Simplifying Process Model Abstraction: Techniques for Generating Model Names

Henrik Leopold[a], Jan Mendling[b], Hajo A. Reijers[c], Marcello La Rosa[d]

[a] *Humboldt-Universität zu Berlin, Germany*
[b] *WU Vienna, Austria*
[c] *Eindhoven University of Technology, Netherlands*
[d] *Queensland University of Technology and NICTA, Australia*

Abstract

The increased adoption of business process management approaches, tools, and practices has led organizations to accumulate large collections of business process models. These collections can easily include from a hundred to a thousand models, especially in the context of multinational corporations or as a result of organizational mergers and acquisitions. A concrete problem is thus how to maintain these large repositories in such a way that their complexity does not hamper their practical usefulness as a means to describe and communicate business operations. This paper proposes a technique to automatically infer suitable names for business process models and fragments thereof. This technique is useful for model abstraction scenarios, as for instance when user-specific views of a repository are required, or as part of a refactoring initiative aimed to simplify the repository's complexity. The technique is grounded in an adaptation of the theory of meaning to the realm of business process models. We implemented the technique in a prototype tool and conducted an extensive evaluation using three process model collections from practice and a case study involving process modelers with different experience.

Keywords: Business process model, model name, process model repository, refactoring, model abstraction

1. Introduction

Recent years have seen a substantial increase in business process modeling initiatives. While process modeling was utilized in the 1990s mainly as a technique for facilitating single process re-engineering efforts [1, 2, 3, 4, 5], many companies have turned to a more encompassing and evolutionary approach to Business Process Management (BPM). This development has led to the establishment of BPM expert teams, competence centers, and consulting departments within organizations. These units typically manage a central repository of business process models capturing various aspects of an organization's business operations.

Email addresses: henrik.leopold@wiwi.hu-berlin.de (Henrik Leopold), jan.mendling@wu.ac.at (Jan Mendling), H.A.Reijers@tue.nl (Hajo A. Reijers), m.larosa@qut.edu.au (Marcello La Rosa)

Process modeling is an *ongoing* activity in a setting that is interwoven with strategic management, quality assurance, controlling, and the specific functional areas of a company. As a result, companies that model business processes often maintain repositories containing hundreds if not thousands of models at a remarkable level of detail [6].

The mass of documentation stored in a process model repository poses considerable challenges for efficient and effective use of these models. At various stages, different stakeholders require individual views on these models for getting an overview, receiving support during their modeling efforts, and for quality assurance purposes. For example, abstract views are a good vehicle for achieving a cognitive fit between the representation of the process and the task at hand, cf. [7, 8]. Moreover, they offer a suitable aggregation of process information, which has been found to be crucial for process improvement initiatives [9].

A survey on how process models are used in practice reveals no less than fifteen different use cases for abstracting and compressing a detailed, fine-granular model towards a smaller model capturing the essential information a stakeholder is after [10]. These use cases clearly motivate the development of techniques that facilitate such transformations, e.g., [11, 12, 13]. These techniques, however, do not address the problem of how to automatically generate a proper *name* for an abstract model. This puts the burden on the stakeholder to come up with a meaningful reference for *each* new abstract model that is derived. This is a task that cannot be ignored, since much of the meaning of a process model can only be derived from what its elements stand for. It is also potentially a highly repetitive task: abstractions for a single process model may be dynamically generated requiring a name proposal each time they are viewed by a stakeholder. The same problem occurs when searching for similar or identical model fragments in large process model repositories. Such "(approximate) clones" can be automatically found [14, 15, 16, 17], but then need to be stored with a meaningful name. An automatic technique for determining the name of a process model clone is not yet available, thus hindering the wider adoption of clone detection in practice.

In this paper, we address the problem of defining names for process models and process model fragments that, for instance, result from applying abstraction or clone identification. Our contribution is an approach that will ease the naming task for users who are interested in creating more abstract views on process models than are readily available to them. Specifically, our approach builds on techniques that analyze the activity labels of a process model, and returns a ranked list of names from which the top ranked name is provided. To this end, we build on insights from theories of meaning and exploratory research to devise naming strategies for process models. As we will argue, the quality of the generated naming suggestions is comparable to those that would be generated by humans; however, in the proposed approach these suggestions are generated in only a fraction of the time a human modeler would need to inspect the underlying (fine-grained) model elements and their interrelations.

We implemented the naming approach in a prototype tool based on natural language processing techniques, and, in order to demonstrate the flexibility of our approach, we configured the tool to detect names for process models specified in two languages: English and German. Next, we validated the approach in various directions. First, we measured the performance and accuracy of our approach using three large datasets from practice. Second, we conducted a case study involving process modelers with varying expertise to evaluate the usefulness and appropriateness of the approach in practice.

The rest of the paper is structured as follows. Section 2 reviews the essential concepts of business process modeling, presents a taxonomy of naming strategies grounded in theories of meaning, and highlights an exploratory study on process model names. Section 3 introduces the approach for automatically deriving process model names. Section 4 discusses the results of the evaluation while Section 5 summarizes achievements, and suggests implications for research and practice. Section 6 concludes the paper with an outlook on future research.

2. Background

In this section, we discuss background work required for our research. First, we present the essential aspects of business process modeling. Next, we identify different techniques for finding a process model name. These techniques form the basis for the approach that is proposed in this paper.

2.1. Business Process Modeling

A *business process* is a collection of inter-related events, activities and decision points that involve a number of actors and objects, and that collectively lead to an outcome that is of value to at least one customer [18]. Business process models capture the essential elements of a business process by means of a diagram. A business process modeling language typically includes elements for defining activities, events and their relationships. *Activities* define units of work which usually take time to be completed, such as filling out a purchase order, while *events* describe things that happen atomically, i.e., they have no duration, such as the receipt of an invoice.

Throughout this paper, we use Event-driven Process Chains (EPCs) as a process modeling language to illustrate our approach. EPCs are a widely used language that captures a process in a graph-based notation [19, 20]. In EPCs, activities are called *functions* and are depicted as rounded boxes. Each activity has a short name, such as “write proposal” or “send invoice.” These names typically use a verb and a corresponding business object. Events are visualized as hexagons in EPCs. There are *start events* to specify when a process starts, *end events* showing the outcomes that can be achieved upon process completion, and *intermediate events* that signal that something happens during the business process, e.g., “invoice is received,” or that certain conditions are met, e.g., “loan amount is greater than 10,000 EUR.”

Finally, the routing elements between activities and events are called *connectors* in EPCs. Routing elements are drawn as circles. An XOR-Split represents an either-or decision, which can be merged by an

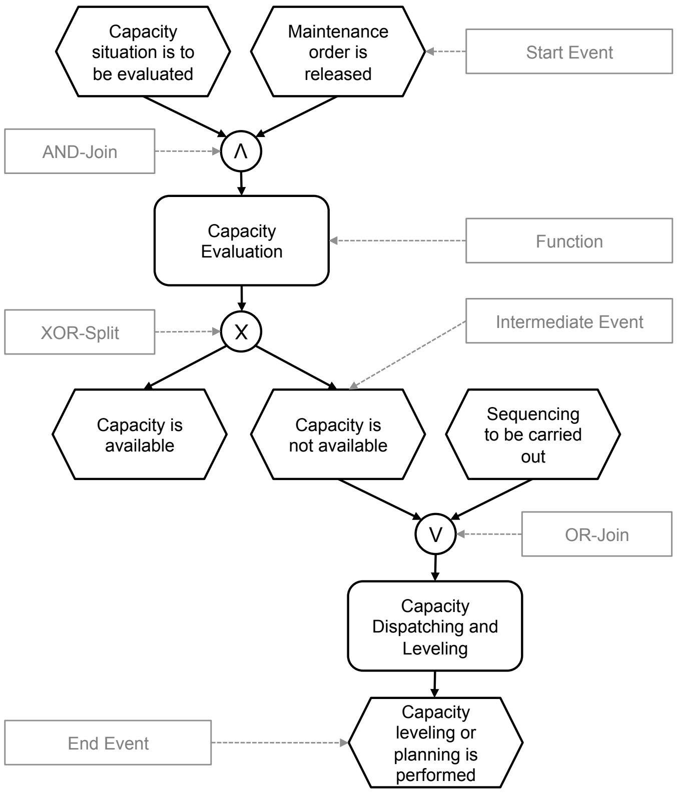

Capacity situation is to be evaluated
Maintenance order is released
Start Event
AND-Join
Λ
Capacity Evaluation
Function
XOR-Split
X
Intermediate Event
Capacity is available
Capacity is not available
Sequencing to be carried out
V
OR-Join
Capacity Dispatching and Leveling
End Event
Capacity leveling or planning is performed

Figure 1: EPC Model "Capacity Planning" from the SAP Reference Model

XOR-join. Both are shown as circles containing an $\times$ symbol. An AND-split defines the point where parallel processing is triggered. The corresponding AND-join synchronizes the concurrent branches. AND elements are depicted as circles with a $\wedge$ symbol. An OR-Split can be used for describing an inclusive choice. The selected branches are synchronized by an OR-join. The symbol of OR elements is a circle with a $\vee$ symbol.

The arcs of an EPC connect these different types of nodes, defining the control flow. While we use EPCs in this article to illustrate our concepts, it must be noted that all ideas can also be applied to other process modeling languages such as BPMN [21], YAWL [22] or UML Activity Diagrams [23].

Figure 1 shows an EPC named "Capacity Planning" from the SAP Reference Model. The process is triggered as soon as the capacity situation needs to be evaluated and the maintenance order is released. After the capacity is evaluated, we observe two possible outcomes. If the capacity is available, the process terminates; otherwise, the capacity is dispatched and leveled again. Once the capacity leveling and planning has been performed, the process is completed.

2.2. Names and Content of Business Process Models

The goal of this section is to clarify the relationship between the content of a business process model and its name. Such a discussion helps to understand the reasons behind naming a process model such as

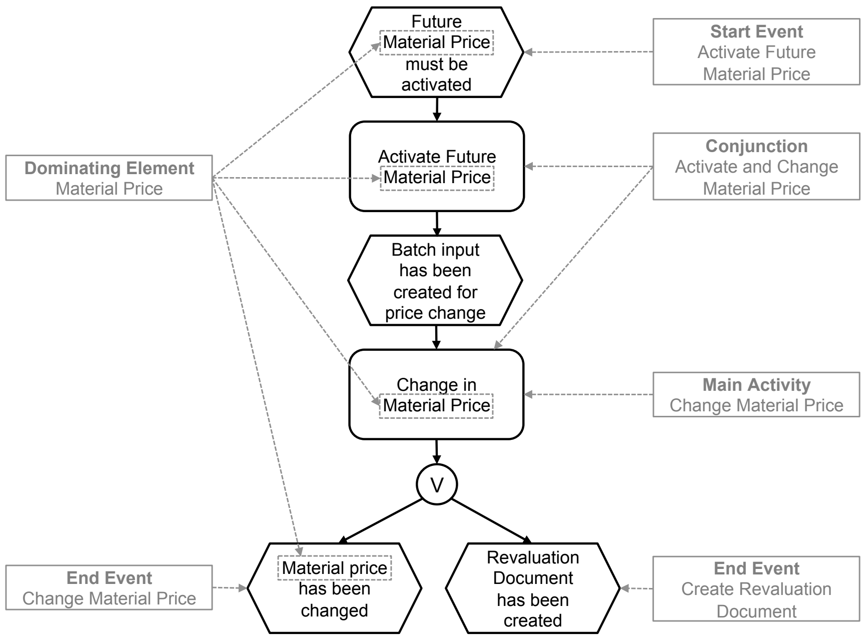


Figure 2: EPC Model "Change in Material Price"

"Capacity Planning." This is a prerequisite in order to able to automatically suggest names for process models or process model fragments.

When humans name complex conceptual entities, such as process models, an intuitive approach seems to point to the core aspect of that entity. However, it is not *a priori* clear what a core aspect is supposed to be. If we can use the meaning of the name of a process to understand its content, there is potentially a way to infer the name from the content. Meaning has become a subject of discourse in philosophy and linguistics with initial work by Lady Welby at the end of the 19th century [24]. A theory of meaning is a concept which tries to determine the meaning of a term or sentence from natural language [25, pp. 1-3]. The benefits of such theories for information systems research have recently been demonstrated by Evermann [26], who clarifies the foundations of schema matching.

In the frame of theories of meaning, we have to ask the question why a person might judge a name such as "Capacity Planning" to capture the meaning of the process model shown in Figure 1. There are two kinds of theories of meaning that approach this question differently [27]: semantic theories defining meaning based on a system references and foundational theories of meaning focusing on the intention of the speaker. In this paper, we refer to both kinds of theories, and focus on those aspects that can be related to process models. In particular, we discuss five perspectives, which are grounded in the feature theory of meaning, prototype theory of meaning, early and late pragmatist theory of meaning, and knowledge-based theory of meaning.

For this discussion, we focus on a process model as a composite 'thing' [28], of which we know its start and end events as much as the activities composing it. We relate each theory to naming strategies. We will use the “Change in Material Price” process model from Figure 2 to illustrate how the different theories of meaning are related to process names.

Feature theory of meaning. The feature theory of meaning relates to the classical view of meaning as a semantic theory on how propositions are evaluated [29, 30, 31]. In this context, terms can be resolved stepwise from their syntactical form to a reference to something in the real world. The way of resolution can be expressed using logical operators, predicates, and referents. A term can then be defined based on a sufficient number of propositions that are true to distinguish it from other ones. For instance, if we aim to distinguish the term “Asset Activation” from “Plant Maintenance” as process names, we might have to consider propositions such as “involves year closing operations” or “involves maintenance order printing.” This means that a model would acquire its meaning through those events and activities which sufficiently describe its overall semantics. Thus, the verbs and business objects used in the activities and events’ labels should be closely related to the name of the process model. In previous work [32] we have observed that this type of naming is used in practice in two variants: either a repeatedly mentioned business object or verb is used to build the name (*dominating element*) or a conjunction of several activities (*conjunction of activities*). Following these strategies, our example process model from Figure 2 would carry a name that includes the dominating business object “Material Price.” In the more specific case of the conjunction strategy, it would be accordingly named as “Activate and Change Material Price.”

Prototype theory of meaning. The feature theory of meaning has certain weaknesses in dealing with atypical sub-categories. For example, an amphibious vehicle can swim, while a vehicle in general cannot. Rosch [33, 34] approached this issue by stating that a term can best be defined based on one prototypical instance. In line with this argument, the term “cooking” often stands for a set of activities that includes preparing and mixing the ingredients. However, in many cases cooking is chosen to represent the overall process. Such a strategy would be applicable to process models where one particular activity represents the core of the process. Other activities may have the character of preparatory or closing activities which contribute to this core activity. This naming strategy is apparent from the discussion in [35]. Here, the authors describe the notion of a ‘core of a process.’ This core process covers the essence of a process in terms of those steps that are relevant for an external party to interact with it. In the example model, a main activity is given by the “Change in Material Price” function, which could be used to name the model according to this strategy.

Early pragmatist theory of meaning. The above mentioned theories are rooted in propositional concepts. Pragmatist theories, by contrast, put an emphasis on observable effects of words and sentences [36, pp. 5-10], [37, pp. 184-186]. For instance, the meaning of a stop sign is more centrally related to its ability to create a certain effect than to its features of being red. We would thus expect that the effect of a process model called "Asset Activation" or "Plant Maintenance" would be that "asset activation is completed" or "plant maintenance is completed," respectively.

When we apply this perspective to process models in general, we have to focus on those parts where effects of executing the process are explicitly mentioned. These are typically the process model's end events. Indeed, this line of thinking is found in existing work that is related to the problem at hand. For example, Sharp and McDermott [38, p.80] suggest identifying a process from its outcomes. Applying this idea to the example model, we could use the left end event to name the process model. From this event, we could then derive the name "Change in Material Price."

Late pragmatist theory of meaning. There are certain issues with the early pragmatist view of meaning. To illustrate this with respect to an example used earlier, one can imagine that a driver does not always obey a stop sign. If this actual effect is essential to its meaning, it might not be considered to be a stop sign anymore. This problem is resolved by the late pragmatist theory. It states that the meaning of a word or phrase is not its actual but its *intended* effect in the real world [39, pp. 67-82], [40, pp. 213-224]. Accordingly, a stop sign is still a stop sign even if it is not obeyed, because it intends to make one stop. This perspective implies that we have to identify places in a process model that might potentially point to its intention. Such an intention may be made explicit in the label of start events. For example, a label like "Future Material Price must be activated" used for the start event of the model in Figure 2 suggests what the process model is going to be about, i.e., the activation of the future material price. From this label we can thus infer the "Activate Future Material Price" for the process model at hand. Such an approach would be consistent with modeling approaches that closely relate processes to achieving business goals [41, 42].

Knowledge-based theory of meaning. The knowledge-based theory of meaning emphasizes that terms cannot be fully understood with the theories mentioned above. A knowledge theory builds on a causal network that relates different concepts. Via an act of baptism [43], an a-priori arbitrary term is associated with a concept. Later on, the usage of the term becomes historical and embedded. For instance, the term "photosynthesis" is a composite term that refers to the process of converting carbon dioxide into organic material that requires sunlight and produces oxygen. Knowledge-based names are also used for business processes. Often, they cannot be directly traced back to terms appearing in a process model. Such cases can be partially related to a process being a composite thing, which as a system possesses emergent properties [28]. This phenomenon may be present in production processes when the to-be-constructed entity is not explicitly mentioned. For instance, the approach by Reijers et al. [44] suggests deriving process models from a product-based decomposition model. In this case, all activities of the process model refer to sub-components of the to-be-constructed object. For instance, if we introduced the name "Adjustment in Selling Price" for the model in Figure 2, the name would still reflect the contents of the model but only partially refer to the terms appearing in the model's labels.

Table 1: Mapping of Naming Strategies to Automated Sub-Approaches

Theory of Meaning	**Information Source**
Feature Theory of Meaning	Multiple Activities
Prototype Theory of Meaning	Main Activity
Late Pragmatist Theory of Meaning	Start Event
Early Pragmatist Theory of Meaning	End Event
Knowledge-based Theory of Meaning	Model Semantics

3. Derivation of Name Proposal

The different theories of meaning exhibit relative strengths and weaknesses. Semantic theories are typically subject to a context of utterance reflecting the circumstances of evaluation, and to modes of presentation. Foundational theories emphasize the importance of: the speakers' meaning and intentions, beliefs, and social norms. Thus, names of process models will have different meanings in different settings, just as the process of registering a new-born child will hardly be exactly the same in two municipalities, cf. [45]. While the content can differ, it may also be the case that synonymous process names are used. In this vein, many authors emphasize the importance of naming conventions [46, 47, 6]. The most dramatic statement to reflect this fact is made by a process modeler cited by Recker et al. [48] saying "Certainly naming, naming is one of our horrible, horrible challenges."

The aim of this section is to define an automatic approach for the derivation of a name proposal that is consistent with the content of a process model. We align our approach with an informal concept for name derivation described in [32]. First, we identify the requirements for the definition of an approach for automatic name generation. Next, we present the algorithms we defined for automatically naming process models, after which we present the approach for ranking the name proposals from the generated list.

3.1. Requirements for Generating Process Names

As pointed out in the previous section, most naming concepts directly refer to the textual information captured in the process model labels. Hence, in order to automatically generate names for process models, it is necessary to adequately infer this information from the process model labels. Table 1 provides an overview of the introduced theories of meaning and the associated information source within the process model.

The label analysis and the related extraction of textual information is impeded by two particular properties of activity labels. First, activity labels are very short and typically do not contain proper grammatical sentences. Hence, they provide very limited information about the grammatical role of individual words. Further, many activity labels suffer from the zero-derivation phenomenon [49]. While some English verbs are transformed into a noun by adding suffixes as "ize" or "(i)fy", many words represent verbs and nouns without any syntactical changes. As an example, consider the noun "the order" and the verb "to order." This zero-derivation property is a significant challenge in the context of label analysis. A a result of these properties, the standard application of natural language processing tools such as the Stanford Parser is not applicable in the context of activity labels.

In order to overcome these challenges, we employ the label parsing technique proposed in [50]. This approach builds on the insight that the majority of activity labels follow regular structures, so-called activity label styles [51]. Based on this observation, labels containing an action are classified into three different label styles: verb-object style, action-noun style, and the descriptive style. In verb-object labels, the action is given as an imperative verb in the beginning, followed by a business object. Examples are "Sign Contract" or "Order Materials." Action-noun labels do not contain a verb since the action is provided as a noun, as in, for example,"Creation of Invoice and "Contract Verification." In descriptive labels, the task is described from a third person perspective. Many of these labels start with the role executing the activity. The role is followed by the action in the third person form and the business object. Examples of descriptive labels are "Customer signs Contract" and "Checks Invoice." Once a given label is assigned to one of these label styles, the structure of the corresponding label style can be used to infer the action and the business object from the label.

Utilizing the structural insights from these label structures, the context of the activity is considered to assign it to one of these label styles. Thereby, the context is structured into four levels: 1) activity label itself, 2) process model containing the activity, 3) process model collection, and 4) knowledge on word frequencies. The strategy of the algorithm is to first use the most local context in order to classify the considered label. Once the previous context level turns out to be insufficient, the scope is broadened. As a result, each label can be classified, and the action and business object for each function can be reliably determined. Having the textual information from the process model at hand, we can then use this information for automatically generating process names.

3.2. Automatic Approach to Generate Process Names

In this section, we present our automatic technique for generating process model names. It builds on the theoretical concepts discussed in Section 2.2 and the labeling practices from Section 3.1. The fundamental idea of the approach is the generation of a set of potentially useful names for a given process model. Afterwards, the name proposals are ranked according to their appropriateness.

Our technique is organized in three phases, as illustrated in Figure 3. Phase 1 serves as a preparation step. We make use of the analysis technique introduced in Section 3.1 in order to automatically annotate all activities and events with their action and business object. We extended this technique with respect to its ability to analyze start and end events as defined in [52]. The second phase represents the main step of our approach, consisting of a set of different techniques to generate name proposals. In the middle part of Figure 3 each technique is depicted as a rectangle. Finally, in the third phase, the single best (or the k-best) name proposals are selected and transformed to the verb-object style in order to present an understandable and

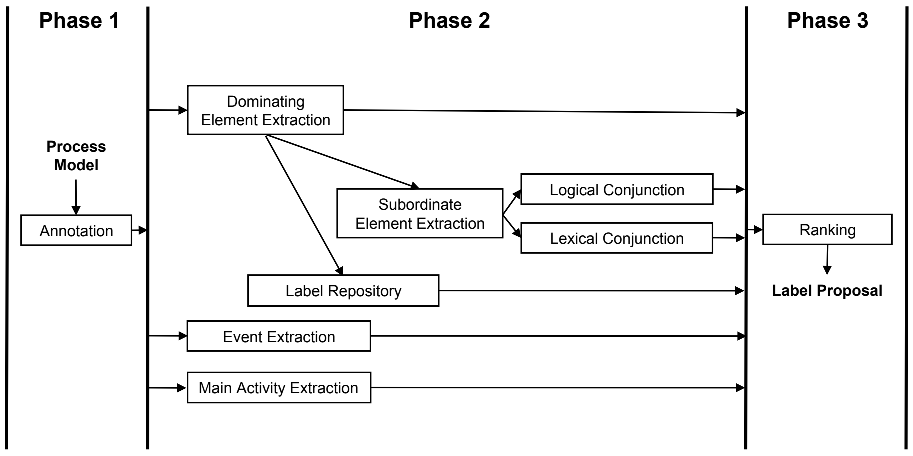


Figure 3: Overview of the Name Generation Approach

Algorithm 1: Dominating Element Extraction

```
1:  extractDominatingElements(ProcessModel $model$, String $type$)
2:  List $elementCount$ = new List();
3:  List $elements$ = $model$.getElements(type);
4:  for all elements $elem$ in $model$ do
5:      $currentCount$ = elementCount.get($elem$);
6:      $elementCount$.set($elem$,$currentCount$+1);
7:  $maxCount$ = $elementCount$.getMax();
8:  if amount of elements $elem$ with count $maxCount$=1 then
9:      return $elem$ with count = $maxCount$;
10: else
11:     return “”;
```

unambiguous name.

In the remainder of Section 3.2, we introduce each of the name generation techniques from Phase 2 in greater detail and explain their interdependencies.

3.2.1. Dominating Element Extraction

The *Dominating Element Extraction* builds on the insights from the feature theory of meaning. The goal of this technique is to identify whether the given process model includes a dominating action or a dominating business object. Therefore, the occurrence of each action and business object among all activities in the model is determined. Due to the activity annotation from Phase 1, this step is straightforward.

Algorithm 1 formalizes the details of the dominating element extraction. It requires two input parameters: the considered process model and the variable type, which determines whether the algorithm is supposed to search for a dominating action or a dominating business object. The main part of the algorithm scans the activities and checks the occurrence of each action or business object (lines 4-7). If one action or business object has a higher occurrence than do all other elements, it is saved as the dominating element (lines 8-9). If the process model contains such a dominating term, it can be used as input for the *Subordinate*

Algorithm 2: Subordinate Element Extraction

```
1: extractSubordinateElements(List $activityList$, String $dominatingElem$, String $type$)
2: List $subordinateElements$ = new List();
3: for all activities $a$ in $activitiyList$ do
4:     if $a$ contains $dominatingElem$ then
5:         if $type$ = “Action” then
6:             $subordinateElements$.add($a$.getBusinessObject());
7:         else
8:             $subordinateElements$.add($a$.getAction());
9: return $subordinateElements$;
```

Element Extraction technique. Otherwise, the approach moves on to the *Event Extraction* and the *Main Activity Extraction* techniques, as they do not require the input of a dominating element. As an example of a dominating element, we reconsider the process model “Change in Material Price” from Figure 2. In this process model, both activities contain the business object “Material Price.” Accordingly, this business object is determined as the dominating element.

3.2.2. Subordinate Element Extraction

The *Subordinate Element Extraction* derives a list of terms which is processed in the context of two further techniques: the *Lexical Conjunction* and the *Logical Conjunction.* More specifically, the Subordinate Element Extraction identifies all terms which co-occur with the dominating element. If, for instance, the dominating action “confirm” was derived from the two activities “Order Confirmation” and “Contract Confirmation,” the subordinate elements are given by the business objects “order” and “contract.” Clearly, if the dominating element is an action, all subordinate elements will be business objects and vice versa.

Algorithm 2 illustrates this procedure. The algorithm requires three parameters: a list of all activities from the considered process model, the corresponding dominating element, and the type variable, which determines whether the dominating element is an action or a business object. As a result, it returns a list of subordinate elements. Therefore, each label in the activity list is inspected whether it contains the dominating element (line 4). If this is the case, the relevant subordinate element is extracted and stored in the result list *subordinateElements* (lines 6,8). After all activities have been investigated, the result list is returned (line 9).

3.2.3. Lexical and Logical Conjunction

Based on the concept of the feature theory of meaning, we introduce two different conjunction techniques: *Lexical Conjunction* and *Logical Conjunction.* The general goal of these techniques is to find an adequate composition of the dominating element and the derived subordinate elements.

The Lexical Conjunction implies the replacement of the subordinate elements with a newly introduced term derived from the lexical relations among these elements. In our context, we may make use of two lexical relations: *holonyms* (a word representing the whole of a part-of relation) and *hypernyms* (a more

Algorithm 3: Lexical Conjunction

```
1: getLexicalConjunctions(List subordinateElems, String dominatingElem, String type)
2: lexicalConjunctions = new List();
3: subordinateElemsClean = new List();
4: for all elements elem in subordinateElems do
5:     if WordNet contains entry for elem as type then
6:         subordinateElemsClean.add(elem);
7: if subordinateElemsClean.length > 1 then
8:     holonyms = subordinateElemsClean.get(1).getHolonyms();
9:     hypernyms = subordinateElemsClean.get(1).getHypernyms();
10:    for i=2 to subordinateElemsClean.getLength() do
11:        holonyms = getCommonElems(holonyms, subordinateElemsClean.get(i).getHolonyms());
12:        hypernyms = getCommonElems(hypernyms, subordinateElemsClean.get(i).getHypernyms());
13:    lexicalConjunctions.add(holonyms);
14:    lexicalConjunctions.add(hypernyms);
15: returnList = new List();
16: for all conjunctions c in lexicalConjunctions do
17:     returnList.add(new ProcessName(dominatingElem, c, type));
18: return returnList;
```

general word). Hence, we employ WordNet to identify common holonyms and hypernyms for the given subordinate elements. If such a word is found, a process name is composed using the dominating element and the identified holonym or hypernym. As an example, consider the actions “check” and “review.” Using the WordNet database, we can identify the hypernym “analyze” which semantically covers both words and can hence be employed for replacing them.

Algorithm 3 illustrates the details of this technique. It requires three input parameters: a list of subordinate elements, the dominating element and the type variable, which determines whether the dominating element is an action or a business object. The algorithm returns a list of process names, as constructed from the identified holonyms or hypernyms and the dominating element.

In the beginning of the algorithm, the original subordinate element list is reduced to those elements which can be found in the WordNet dictionary (lines 4-6). As we use WordNet for deriving lexical relations between words, this is an inevitable step. In case more than one single element is left, the common hypernyms and holonyms for these elements are identified. Therefore, the set of all hypernyms and holonyms for the first element in the *subordinateElementsClean* is determined (lines 8-9). Subsequently, these two sets are reduced to those hypernyms and holonyms which are shared by all considered subordinate elements (lines 10-12). In this way, those holonyms and hypernyms are identified which all subordinate elements have in common. Afterwards, for each identified holonym and hypernym, a process name with the dominating element is constructed. The resulting process names are stored in a list (lines 15-17). Finally, the name list is returned.

As opposed to the Lexical Conjunction, the Logical Conjunction does not rely on an existing lexical relationship among the subordinate elements. The Logical Conjunction simply uses the logical operators “and” or “or” to connect the subordinate elements. As a result, business objects belonging to semantically different domains such as “Contract” and “Information System” can also be combined in a process name, complemented by the dominating action.

Algorithm 4: Derivation from Label Repository

```
 1: deriveFromLabelRepository(List repository, String dominatingElem, String type, Integer maxCount)
 2: List candidates = new List();
 3: List candidateCount = new List();
 4: for all activities a in repository do
 5:     if a.getElement(type) = dominatingElement then
 6:         if candidates contains a then
 7:             currentCount = candidateCount.get(a);
 8:             candidateCount.set(a,currentCount+1);
 9:         else
10:             candidates.add(a);
11:             candidateCount.add(a.toProcessName(),1);
12: order candidates by candidateCount;
13: if candidates.getLength() > maxCount then
14:     candidates = candidates.getFirstElements(maxCount);
15: return candidates;
```

3.2.4. Label Repository

The *Label Repository* approach aims to generate process names that include information from other process models. Thus, this technique is related to the knowledge-based theory of meaning. However, as it also relies on the identified dominating elements, it is associated with the feature theory of meaning as well. In order to generate process names the Label Repository approach employs the activities of all available process models to construct a so-called label repository, a huge list of activities. Using a dominating element, this repository can be consulted to find complementing elements that are likely to co-occur with the dominating element. This strategy is based on the observation that some actions are regularly associated with a typical set of business objects and vice versa.

As an example, consider the process "Capacity Planning" from the SAP Reference Model depicted in Figure 1. Now assume that we would like to find a name for this process. Using the Dominating Element Extraction, we can identify the dominating business object "Capacity." As this is apparently not a very comprehensive name, we employ the Label Repository approach to find the most frequent terms in the repository which are related to the dominating element. We then compose the dominating element with these terms to generate according process model names. From the SAP Reference Model we can build a considerable label repository containing 2,433 activities. By looking up the business object "Capacity," we obtain, amongst others, the process name "Capacity Planning," which completely matches the SAP process name. Thus, the Label Repository approach can be used to turn a non-comprehensive proposal from the Dominating Element Extraction into an appropriate process name.

Algorithm 4 provides a formal description of the Label Repository approach. The algorithm requires four input parameters: a list of activities which is used as the label repository, the dominating element of the considered process model, the type of the dominating element, and the maximum number of elements which may be returned. The output of the algorithm is a list of name proposals. At the start of the algorithm the two lists *candidates* and *candidateCount* are created (lines 2-3). In this way, the list *candidates* serves as a

Algorithm 5: Event Extraction

```
1: extractFromEvents(List eventList, String eventType)
2: returnList = new List();
3: for all events e in eventList do
4:     if eventType = "Start Event" then
5:         if e contains "required" or "is necessary" or "must be" or "needs to be" or "to be" then
6:             returnList.add(e);
7:     if eventType = "End Event" then
8:         if e contains "executed" or "carried out" or "performed" or "was" or "were" then
9:             returnList.add(e);
10: for all events e in eventList do
11:     returnList.add(e.toProcessName()));
12: return returnList;
```

storage for the identified candidate activities and the list *candidatesCount* is employed for saving the count of each of these candidates. In the subsequent loop all activities are scanned for the dominating element (lines 4-5). If the dominating element is found, it is looked up in the candidate list. In case it already exists in the list, the index is determined and the count in *candidateCount* is increased accordingly (lines 7-8). Otherwise, if the activity is not already stored, it is saved to the candidate list and its count is set to *1* (lines 10-11). Once all activities in the repository have been investigated, the candidates are ordered with respect to their count (line 12). If the number of candidates exceeds the maximum size, given by *maxCount*, the list is shortened to the first and thus most frequent elements (lines 13-14). Finally, the candidate list is returned (line 15).

3.2.5. Event Extraction

The goal of the *Event Extraction* technique is the generation of process names from start and end events. Hence, it builds on the early and late pragmatist theories of meaning. The generation of names from start and end event consists of three steps. First, start and end events must be automatically recognized. Therefore, all events are anaylzed with regard to their position in the model. If an event has no predecessor, it is categorized as a start event. If an event has no successors, it is categorized as an end event. Second, the identified start and end events are analyzed with regard to their merit to provide useful information about the process model. For instance, the term "was" in the start event "Asset was found" is more likely to represent a condition for triggering the process than actually indicating what the process is dealing with. By contrast, in the start event "Asset is to be created" the term "is to be" clearly points to an action which is covered by the process. Hence, "Create Asset" may represent a suitable process name. We derived an extensive classification of these signalizing terms from the investigated process model collections. As a result, this decision can be made in an automated fashion. As a last step, process names are derived from actions and business objects of the relevant events.

Algorithm 5 formalizes this approach. It requires a list of events of the considered process model and the corresponding event type. As a result, the algorithm provides a list of potentially useful process names. In

Algorithm 6: Main Activity Extraction

```
1: extractMainActivities(List $activityList$)
2: $returnList$ = new List();
3: for all activities $a$ in $activityList$ do
4:     $precedingEvents$ = $a$.getPredecessors();
5:     $firstActivity$ = true;
6:     for all events $e$ in precedingEvents do
7:         if $e$.getPredecessors().getLength() $> 0$ then
8:             $firstActivity$ = false;
9:     if $firstActivity$ = false then
10:        $succeedingEvents$ = $a$.getSuccessors();
11:        $lastActivity$ = true;
12:        for all events $e$ in succeedingEvents do
13:            if $e$.getSuccessors().getLength() $> 0$ then
14:                $lastActivity$ = false;
15:    if $firstActivity$ = true or $lastActivity$ = true then
16:        $returnList$.add($a$.toProcessName());
17: return $returnList$;
```

order to obtain a high number of useful process names, all events in the *eventList* are scanned for signal constructs. Accordingly, it is differentiated between start and end events (lines 4,7). As illustrated in the example in the preceding paragraph, start events are checked for constructs indicating that something is going to be performed (line 5) and end events are inspected for constructs indicating that something was conducted during the process (line 8). If one of these constructs is found, the event is added to the *resultList* (lines 6,9), which is subsequently used for constructing process names (lines 10-11). In the last step, these names are returned (line 12).

3.2.6. Main Activity Extraction

By referring to the prototype theory of meaning, this technique tries to automatically identify main activities. As this requires determining the role of activities in the process model, this is a complex task. In order to overcome this problem, we utilize the insights of a comprehensive analysis of three industry process model collections. From this analysis we learned that approximately 85% of all main activities are positioned either at the start or at the end of a process. For example, imagine a process handling the shipping of a good. Such a process would typically include a set of preparatory steps before the last activity eventually asks for the actual shipping. Presuming the existence of a main activity, our approach uses the first and last activities of a process to generate corresponding name proposals.

Algorithm 6 illustrates the required steps. The algorithm expects a list of all activities from the process model and returns a list of process names derived from the potential main activities. As the algorithm assumes the existence of main activities, the core task is the identification of first and last activities. Therefore, all activities from the *activityList* are investigated as follows. First, the preceding events of the considered activity a are determined and the variable $firstActivity$ is set to *true* (lines 4-5). Subsequently, all identified events are checked for predecessors (lines 6-7). In case preceding activities are found for any of these events,

Algorithm 7: Main Algorithm Phase 2

```
1: performMainAnalysis(ProcessModel model, List repository)
2: returnList = new List();
3: dominatingAction = extractDominatingElements(model, “Action”);
4: dominatingBusinessObject = extractDominatingElements(model, “Business Object”);
5: if dominatingAction.hasValue() and dominatingBusinessObject.hasValue() then
6:     returnList.add(new ProcessName(dominatingAction, dominatingBusinessObject));
7: if dominatingAction.hasValue() then
8:     dominatingElem = dominatingAction;
9:     subElems = extractSubordinateElements(model.getActivities(), dominatingElem, “Action”);
10:    returnList.add(performLexicalConjunction(subElems, dominatingElem, “Action”));
11:    returnList.add(performLogicalConjunction(subElems, dominatingElem, “Action”, “and”));
12:    returnList.add(deriveFromLabelRepository(repository, dominatingElem, “Action”, 5));
13: if dominatingBusinessObject.hasValue() then
14:    dominatingElem = dominatingBusinessObject;
15:    subElems = extractSubordinateElements(model.getActivities(), dominatingElem, “BO”);
16:    returnList.add(performLexicalConjunction(subElems, dominatingElem, “BO”));
17:    returnList.add(performLogicalConjunction(subElems, dominatingElem, “BO”, “and”));
18:    returnList.add(deriveFromLabelRepository(repository, dominatingElem, “BO”, 5));
19: returnList.add(extractFromEvents(model.getEvents(), “Start Event”));
20: returnList.add(extractFromEvents(model.getEvents(), “End Event”));
21: returnList.add(extractMainActivities(model.getActivities()));
22: return returnList;
```

the activity a cannot be a first activity of the process model. Hence, $firstActivity$ is set to $false$ (line 8). By contrast, if none of these events has preceding activities, $firstActivity$ accordingly remains *true*. If the activity did not turn out to be a first activity, the analogous steps are performed for determining whether a is a last activity (lines 10-14). Finally, if the considered activity was identified to be a last or a first activity, it is added to the result list (line 16) which is subsequently returned (line 17).

3.2.7. Combining the Naming Techniques

In order to obtain a single but all-encompassing approach, we combine all techniques presented in the previous section illustrated in Figure 3. The execution order of the different techniques is given by their interdependencies. Hence, the Dominating Element Extraction must be executed prior to the Subordinate Element Extraction since the latter requires the dominating element as input. Accordingly, the Subordinate Element Extraction must be performed before the Lexical and Logical Conjunction techniques can be applied. However, as the Main Activity Extraction and the Event Extraction are acting independently of other techniques, they can be performed at any time.

Algorithm 7 illustrates the composition of the main approach. It requires an annotated process model from Phase 1 and the label repository list as input, after which it returns a list of name proposals. The first step of the algorithm is the determination of a dominating action and a dominating business object (lines 3-4). In case both dominating elements exist, a name proposal is derived from their combination (lines 5-6). The subsequent steps are dependent only on one dominating element. However, if both dominating elements exist, the proposals for both types are generated. The algorithm proceeds by checking for a dominating activity (line 7). If a dominating activity exists, it is stored in a variable (line 8). Subsequently, the subordinate elements are extracted (line 9). On the basis of these, the Lexical Conjunction, the Logical Conjunction and the Label Repository techniques are performed (lines 10-12). In order to limit the number of names returned by the label repository technique, its threshold is determined with five. This number can be adjusted accordingly, depending on the application scenario. In case a dominating business object exists, the introduced steps are executed accordingly (lines 13-18). Then, the three independent techniques, the Main Activity as well as the Start and End Event Extraction are executed (lines 19-21). Once all techniques have been employed, the ranking of the name proposals in Phase 3 is triggered.

3.3. Ranking of Name Proposal

This section introduces our approach for automatically ranking the name proposals. In order to accomplish this, it is necessary to quantify the appropriateness of the names in the proposal list. As we aim for ranking the names according to how well they reflect the semantics of the model, we introduce an approach for computing the semantic closeness between a name proposal and a process model. The rationale of this approach is the semantic comparison of the components, i.e., action, business object and additional information fragment of a name proposal and the considered process model. To calculate the semantic similarity, we make use of the taxonomy WordNet. A variety of different proposals for calculating the similarity between two concepts based on taxonomies have been introduced in the past [53, 54, 55]. Here, we employ the similarity measure introduced by Lin, as it has been shown to correlate well with human judgements [56].

For calculating the semantic closeness between a name proposal label l^n and a process model m, we introduce four functions: a component similarity function sim_c, a coverage function cov, a label similarity function sim_l, combining the latter two to label similarity result, and a semantic closeness function sc, which uses the function sim_l to determine the overall semantic closeness.

The function sim_c calculates the semantic similarity between a component of the process name label l_c^n and a process model label l_c^m. In general, the function returns the result of the Lin measurement. In case neither of the two labels contains the component, the value is set to zero.

$$sim_c(l_c^n, l_c^m) = \begin{cases} 0 & \text{if } l_c^n = \emptyset \vee l_c^m = \emptyset \\ Lin(l_c^n, l_c^m) & \text{if } l_c^n \neq \emptyset \wedge l_c^m \neq \emptyset \end{cases} \tag{1}$$

The coverage function cov is used to determine the number of components in a label l. Assuming that a label at least refers to an action, the result of cov ranges from 1 to 3. Note that the index a in the definition denotes the action, bo the business object and add the additional information fragment.

$$cov(l) = \begin{cases} 1 & \text{if } l_a \neq \emptyset \wedge l_{bo} = \emptyset \wedge l_{add} = \emptyset \\ 2 & \text{if } l_a \neq \emptyset \wedge (l_{bo} \neq \emptyset \veebar l_{add} \neq \emptyset) \\ 3 & \text{if } l_a \neq \emptyset \wedge l_{bo} \neq \emptyset \wedge l_{add} \neq \emptyset \end{cases} \tag{2}$$

In order to combine the individual component similarity results, we introduce the function sim_l. This function calculates the arithmetic mean of the similarity values for action, business object and the additional information. This is accomplished by dividing the sum of sim_a, sim_{bo} and sim_{add} by the maximum coverage among l^n_c and l^m_c. As a result, we obtain the overall semantic similarity between a name label and a process model label.

$$sim_l(l^n, l^m) = \frac{sim_a(l^n, l^m) + sim_{bo}(l^n, l^m) + sim_{add}(l^n, l^m)}{\underset{l \,\in\, \{l^n, l^m\}}{\arg\max}\; cov(l)} \tag{3}$$

By calculating the average value of sim_l using the set of all element labels L^m comprised in the process model m, we obtain the overall semantic closeness sc of a considered process name l^n.

$$sc(l^n, m) = \frac{\sum\limits_{l^m_i \in L^m} sim_l(l^n, l^m_i)}{|L^m|} \tag{4}$$

Comparing the semantic closeness of all generated name proposals, we rank the name proposals according to their score. In scenarios where a single best name has to be shown automatically, only the best ranked proposal is returned. In more interactive scenarios, a list of k-best proposals can be shown.

4. Evaluation

To demonstrate the capability of our approach to find appropriate process model names, we conducted a two-part evaluation. First, we conducted an experiment using our prototype implementation and different process model collections from practice in order to demonstrate the efficiency of the implementation and the semantic closeness of the proposed model names to the original names. Second, we ran a case study with a large insurance company to reflect on the usefulness and appropriateness of the approach in a business context. In both settings, we configured our technique to return the top-ranked proposal only.

4.1. Experiment

For the experiment, we used three process model collections from practice with different characteristics including: domain, model size, naming strategies, and the natural language used in the labels. These collections are:

- The **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 [57, pp. 145-164]. The collection contains a total of 604 EPCs organized in 29 functional branches of an enterprise such as sales and accounting.

Table 2: Characteristics of the three process model collections

Property	**SAP**	**CH**	**IM**
Process Models	604	328	88
Activies	2433	4414	293
Events	6948	10292	584
Average No. of Activities per Model	4.0	13.5	3.3
Average No. of Events per Model	11.5	31.4	6.6
Language	English	English	German
Dominating Element Naming	22%	2%	0%
Main Activity Naming	22%	29 %	51%
Start / End Event Naming	9%	22%	40%
Dominating Element + Semantic Naming	12%	9%	9%
Semantic Naming	35%	38%	10%

- The **Claims Handling Model Collection**: The second model collection contains 328 EPCs dealing with the claims handling (CH) activities of an insurance company. In comparison to the other collections, the insurance model set contains rather large processes with a high density of events.
- The **Incident Management Collection**: This model collection contains the incident management (IM) processes from a large IT service provider. It consists of three abstraction layers and contains in total 88 EPCs. As opposed to the other model collections, the models are generally small and labeled in German.

Table 2 summarizes the main characteristics of the three collections.

With respect to the differences between the three collections, we particularly emphasize three dimensions: model size, naming strategies, and labels language. Considering EPCs with the intention of finding a suitable name, the *model size* is reflected by the number of activities and events in the model. As our approach mostly relies on information which is explicitly given in the model, the number of these elements is an important factor. In case too little information is given it is difficult to provide good proposals. However, if a model contains too many activities or events it becomes challenging to generate a single appropriate proposal. Thus, in order to show the capability of the approach to cover both cases, we included model collections with significantly different average model sizes.

Naming process models is undoubtedly a subjective task which can be conducted in arbitrary ways and may thus lead to varying quality. In order to avoid building on a model set with a specific and homogeneous usage of *naming strategies*, we included model collections with a differing naming style distribution. Thus, both risks, the possibility of encountering alternative naming strategies and the positive or negative effect of one particular naming strategy are mitigated.

Although English is the predominant business language, many companies still tend to model processes in their local language. Recognizing this tendency, we aim at showing that our approach is not restricted to

Table 3: Computation Performance

Figure	**SAP**	**CH**	**IM**
Total Computation Time (ms)	12024	6253	1786
Average Computation Time per Model (ms)	19.91	52.55	20.30
Maximum Computation Time for One Model (ms)	957	312	323

process models with English labels, but also works for other languages. Thus, we included a model collection with German labels.

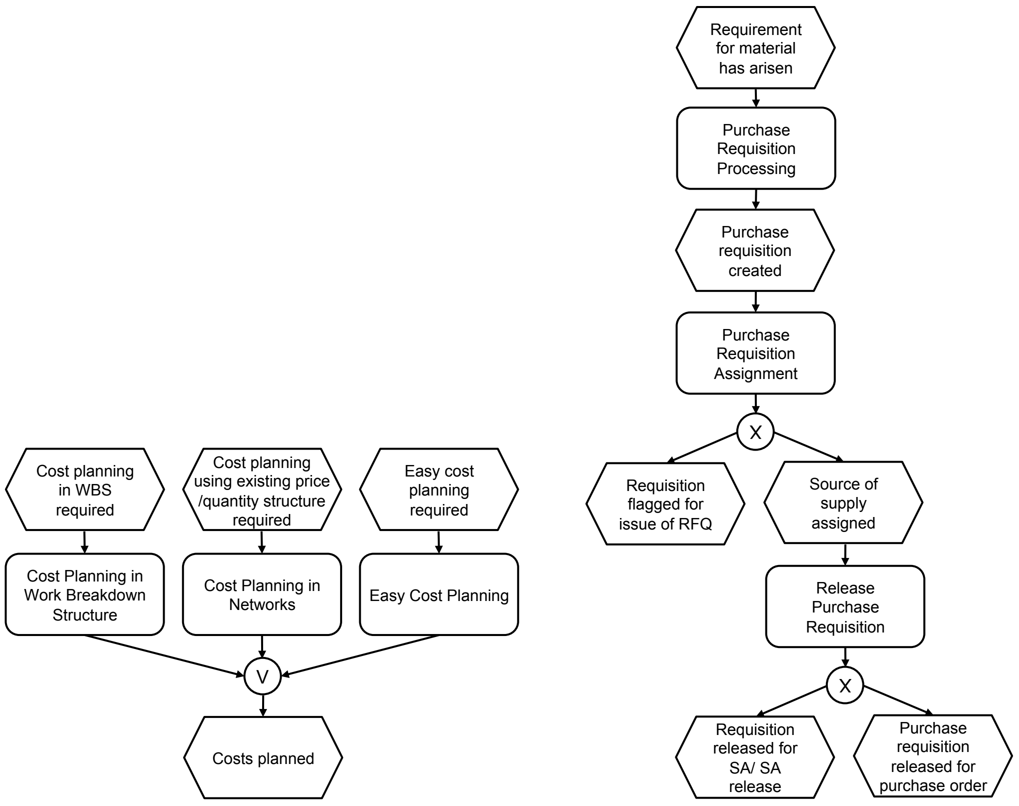
4.1.1. Performance Measurements

Our naming technique was designed to provide support during the modeling process in the context of a modeling tool. Hence, the name proposals should be computed adequately fast. Therefore, we measured the computation times for each of the process models in the considered model collections including the selection of the best candidate. We tested the name proposal generation on a MacBook Pro with a 2.26 GHz Intel Core Duo processor and 4GB RAM, running on Mac OS X 10.6.7 and Java Virtual Machine 1.5. In order to exclude one-off setup times, we ran the technique twice and considered the second run only.

Table 3 summarizes the results of the performance measurement by showing the total computation time for each model collection, the average computation time per model, and the maximum computation time measured for a single model. From these numbers we can draw the following conclusions. First, the results illustrate the applicability of our approach in terms of run-time. Even the longest run for a process from the SAP Reference Model took less than one second. However, considering the average values, this case can be clearly seen as an exception. Second, especially the average computation time for each model shows a clear correlation between run-time and the model size. However, if we take the details of the technique into consideration, it becomes clear that the existence of a dominating element significantly influences the computation effort. Once a dominating element is identified, several other steps - as for instance complex dictionary lookups for lexical relations - are triggered by the conjunction based approaches and the repository consultation. Nevertheless, the observed computation times are very short, hence making our approach suitable for practical applications.

4.1.2. Semantic Closeness and Model Size

We quantify the accuracy of a generated name by measuring its closeness to process model elements. However, as the isolated consideration of the semantic closeness is not sufficient for assessing the quality of the generated names, we compare the generated names with the original names from the process model collections. Accordingly, we employ the semantic closeness function introduced in Section 3.3 to compute the metric sc_{gn} representing the semantic closeness between the generated name and the model, as well as the metric sc_{on} representing the semantic closeness between the original name and the model. Table 4


Cost planning in WBS required
Cost planning using existing price /quantity structure required
Easy cost planning required
Cost Planning in Work Breakdown Structure
Cost Planning in Networks
Easy Cost Planning
V
Costs planned
Requirement for material has arisen
Purchase Requisition Processing
Purchase requisition created
Purchase Requisition Assignment
X
Requisition flagged for issue of RFQ
Source of supply assigned
Release Purchase Requisition
X
Requisition released for SA/ SA release
Purchase requisition released for purchase order

(a) Model named as *Plan costs* (b) Model named as *Handle purchase requisition*

Figure 4: Two SAP models with suitable names but different sc_{gn} values

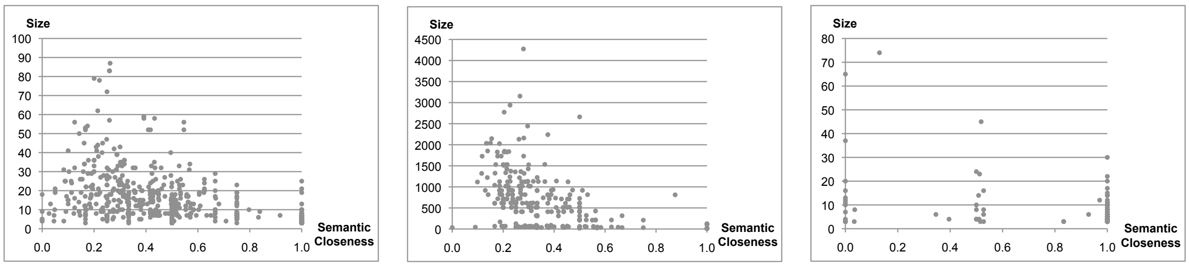
summarizes the results.

The results show that the proposed name generation approach computes names which have, on average, a higher semantic closeness than do the original names. The significant differences in the semantic closeness values between the model collections suggest that the semantic closeness metric is dependent on particular model characteristics. Considering the average number of activities and events per model, it becomes clear that the names of bigger models tend to have a lower semantic closeness. As a single process name cannot fully reflect the semantics of a huge number of model elements, this is an expected result.

The observations concerning model size and heterogeneity is further supported by the data presented in Figure 5. The figures show the connection between the semantic closeness of the generated process names

Table 4: Experiment Results

Model Collection	**Models with** $\mathbf{sc_{gn} > sc_{on}}$	**Models with** $\mathbf{sc_{gn} = sc_{on}}$	**Models with** $\mathbf{sc_{gn} < sc_{on}}$	**Avg.** $\mathbf{sc_{gn}}$	**Avg.** $\mathbf{sc_{on}}$
SAP	501	69	34	0.49	0.21
CH	309	17	2	0.37	0.06
IM	25	57	6	0.93	0.75



(a) SAP Reference Model (b) Claims Handling Collection (c) Incident Management Collection

Figure 5: Disaggregated Naming Results

and the size of the process models for each model collection. The figures reveal that there is an obvious tendency for smaller models to have higher semantic closeness. However, we can also see that this cannot be the only influencing factor, as there are also many small models yielding comparably low closeness values. Investigating the models in detail, it turns out that especially names of models with many different business objects have a small semantic closeness. That, in turn, emphasizes that the semantic closeness value needs to be assessed in light of these model attributes and cannot be assessed independently.

It is now interesting to investigate those cases where the approach works well and where it faces problems. Figure 4 shows two example models with suitable names but different values for sc_{gn}. Figure 4(a) shows a typical example of a model with semantic closeness of *1*. All elements of the models contain the business object "cost" and the action "to plan." Accordingly, the dominating element extraction technique derives these components and combines them to the process name "Plan costs." Figure 4(b) shows a model with more heterogeneously labeled elements. However, as the model contains the dominating business object "purchase requisition," the dominating element extraction technique accordingly identifies this business object as such. Due to the heterogeneity of the comprised actions, the lexical conjunction technique derives the action "handle" as suitable abstraction for the actions of the process model elements. As a result, the name "Handle purchase requisition" is generated. The semantic closeness sc_{gn} for this model is *0.51*.

As an example where the proposed approach fails to generate an appropriate name, let us consider the process model in Figure 6. The interesting characteristic of this model is the handling of two aspects, both described in detail. The main part of the model deals with the long-term planning of the company. Afterwards, once the planning has been conducted, the resulting data are distributed to different departments. Due to the fact that both aspects are mentioned multiple times, the results of the dominating element extraction technique compete with those from the end event extraction technique. In the end, the semantic closeness for *Transfer data* is *0.16*, which is slightly higher than that of the name "Long-term planning." However, this case also illustrates that multiple name proposals would help to adequately solve this problem.

In summary, the semantic closeness comparison indicates that the approach generates useful names which are semantically closer to the models than are the manually created names.

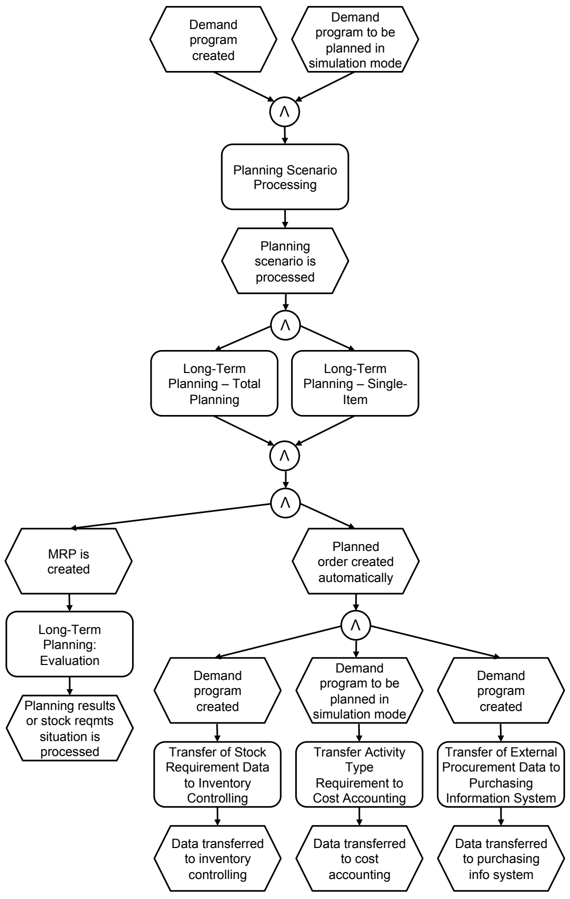

Demand program created
Demand program to be planned in simulation mode
Planning Scenario Processing
Planning scenario is processed
Long-Term Planning – Total Planning
Long-Term Planning – Single-Item
MRP is created
Planned order created automatically
Long-Term Planning: Evaluation
Planning results or stock reqmts situation is processed
Demand program created
Demand program to be planned in simulation mode
Demand program created
Transfer of Stock Requirement Data to Inventory Controlling
Transfer Activity Type Requirement to Cost Accounting
Transfer of External Procurement Data to Purchasing Information System
Data transferred to inventory controlling
Data transferred to cost accounting
Data transferred to purchasing info system

Figure 6: SAP model with non optimal name *Transfer data*

4.2. Case Study

In order to gain insights into the business value of the naming approach, we investigated its usefulness and appropriateness in the context of a case study with a large Australian insurance company. The overall objective was to learn whether experts consider the generated names to be appropriate and how they assess the overall value for their company.

The case study involved four business analysts with modeling experience ranging from two to more than ten years. The case study consisted of two parts. For the first part, we selected ten process model fragments with varying characteristics from the process model collection of this company, and automatically generated process name proposals. These process models had between 3 and 17 activities (7 on average) and between 10 and 21 events (14.1 on average). Based on the results, we designed a report containing the models and the generated names. The participants were asked to review the process models and the generated names and to record their impressions concerning usefulness and appropriateness. They spent approximately 45 minutes for studying the provided report.

In the second part we sought more detailed feedback in the context of an open discussion, which also took about 45 minutes. The main goal was to clarify the following questions:

- How do the participants assess the business value of an approach for automatically proposing names for process models and which use cases do they see?
- Which challenges do the participants perceive in terms of appropriateness and usefulness of the generated names?
- Do the participants have concerns about the approach or do they have recommendations to improve it?

In order to guarantee a proper analysis of the discussion, we audio-recorded the discussion. Altogether, all participants agreed on the effort associated with naming processes or process fragments and confirmed that a tool could greatly ease this task:

> *"It is a laborious exercise to read through the process model, so [this tool] provides a smart way of assessing what are the salient terms of a process."*

The participants were also positive about the *business value* of the tool and confirmed the *use cases* we had in mind. They saw the potential of using the approach for naming process model views and abstractions and also sub-processes that have been extracted from their repository in the context of clone detection. Further, they mentioned the possibility of identifying inconsistent naming in their model collection. In particular, they pointed out:

> *"The tool would be very useful in our company."*

In terms of *appropriateness* and *usefulness*, the participants were generally very positive. However, they identified a specific challenge regarding specific terminology. This challenge is due to the company-specific use of certain terms. Discussing the case of a process model named "Manage Variations" by the naming approach, one participant stated:

> *"It does depend on your exposure to that terminology before, because throughout my time in claims everybody has talked about 'handling a claim' so for me that one ['Manage Variations'] actually didn't go that far off."*

The participants did not have any particular concerns, but provided some suggestions for improvement. First, they suggested generating not only one name but several name proposals to choose from:

> *"If we could have 6-7 terms [name proposals] that would be very useful."*

Our approach works with a ranking based on semantic closeness. While we configured the tool to show the best proposal only for the case study, the selection can be easily extended towards the top three or top five name proposals. As a second point, the experts suggested accompanying the name proposals with a set of key words to 'tag' the process fragment, very similar to tag clouds. One participant stated that he liked:

"... the concept of creating a tag cloud similar to a website, for a process model."

The information that we extract from activities and events can be easily reused for such an exercise, leading to an automatic annotation of the process model. Using the variety of naming approaches which are already implemented in the context of our approach, such a tag cloud could be easily created. The cut-off has to be simply increased from one to any appropriate k to get the k-best proposals as tags. If desired, for instance, only the business objects of these k-best proposals can be used as tags.
The participants also pointed out that considering the context of the repository might have helped to come up with terms not included in the model. For instance, one model did not contain the business object "claim." However, by looking at the context, a human reader would have been able to include that word in the process name.

In general, the findings of the case study showed that experts in the case of the insurance company do not only considered our name proposals to be appropriate, but they also emphasized its usefulness in a business setting. They confirmed the suitability of the proposed approach for the use cases of naming fragments resulting from abstraction or clone identification. We received valuable improvement suggestions along different aspects. As the creation of multiple proposals and a tag cloud is closely related to the naming approach, these suggestions can easily be implemented. The consideration of context in the naming strategy is a valuable direction for future work.

5. Implications

In this section we discuss the implications of our research for research and practice.

5.1. Implications for Research

The findings of this research have implications for process model abstraction, semantic analysis of process models, quality assurance of conceptual models and the management of large process model repositories.

Business process model abstraction is an emerging field in process modeling research that investigates operations on a model that preserve its essential characteristics. There are two basic abstraction operations: elimination and aggregation [58]. Existing approaches typically focus on the question of how the control flow structure can be aggregated, e.g., [59, 11, 12, 13, 60, 61]. Semantic aspects have been taken into account for finding sets of activities which are presumably good candidates for aggregation [62, 63, 64, 65]. A critical assumption of this whole stream of research is that the name for a set of aggregated activities can be easily found. The assumed ease comes with two problems. First, finding a proper name by inspecting a process model is not an instantaneous activity. We know from experiments on process model comprehension, e.g. [66], that reading a model is time-consuming. Second, the assumption ignores the fact that aggregation is relevant to hundreds of process models in process model collections from practice. Browsing process models and obtaining abstracted views is a highly repetitive, but contextual operation [10]. The technique developed in this paper is essential to making process model abstraction work in a practical setting. In this way, it has strong implications for future research in this area.

Semantic analysis of activity labels is a valuable technique for process model matching. This area of research is a sub-discipline of ontology matching [67] requiring automatic techniques for finding correspondences between semantically related activities in two different process models. The identification of such correspondences is a preprocessing step for merging process models [68, 69, 70]. Corresponding techniques take the structure, the behavior and the semantic content of process models into account [71]. Some approaches utilize ontologies for matching labels [72, 73, 74]. Semantic annotations are used as well [75, 76, 77, 78]. The key challenge of process model matching relates to different levels of granularity. Some approaches exist for finding 1:n matches between fine-granular parts in one model and semantically equivalent activities defined at a coarse-granular level in the second model [79, 80, 32]. The technique defined in this paper has the capability of aggregating parts of a fine-granular model, such that matches could more easily be found on a corresponding level of detail. This characteristic might prove beneficial to process model matching, opening up a new direction for tackling the 1:n match problem by translating it into a 1:1 match problem at different levels of abstraction.

The quality of business process models has been recognized as a key factor for process improvement initiatives [81]. Several guidelines and frameworks have been proposed to structure quality assurance [82, 47, 83, 84]. Some of them work on the semantic content of the activity labels in order to check whether the models are compliant with naming conventions such as the verb-object style [51]. Grammar parsers are used to this end by Delfmann et al. [85]. Label analysis is used for finding semantic inconsistencies [86] and for identifying terminological problems [87]. The technique reported in this paper provides the basis for further automatic quality checks. By comparing the derived name with the manually set name, our technique can help to spot inconsistencies between the name of the process and its content. This is highly relevant to conceptual process modeling. To-date, there are various formal analysis techniques available for conducting verification in terms of correctness of a process model. In practice, validation, i.e. checking the consistency between model and reality, is even more relevant, but no automatic techniques exist. Inconsistency between the name of a model and its content might be an indication that the model is incomplete, which would provide at least some support for the validation of process models.

The management of large process model repositories has many facets which are strongly connected with the quality of business process models in general [88]. However, one point which is specifically concerned with repositories is the identification of identical or similar process model fragments. Being able to identify such (approximate) clones and factoring them into separate sub-processes reduces the overall complexity and can greatly ease the overall maintenance. Accordingly, different techniques for the identification of such model fragments have been proposed [16, 14, 17, 15]. The approach presented in this paper complements such techniques by generating a name for each identified fragment. Since fragments have the same properties as complete process models, our approach can be equally applied to process model fragments. Due to the fact that huge model repositories may contain hundreds of identical or similar fragments, the automated naming can effectively support users to quickly assess the content of such fragments.

5.2. Implications for Practice

The results of this research have multiple implications for practice. The developed concepts provide the foundation for corresponding tool support. Our approach has been implemented in a prototype tool and can be integrated in existing process model repository tools such as Apromore [89].[1] As our approach requires start and end events only as much as activities do in order to infer a name, the approach can be used on models defined in other process modeling languages beyond EPCs, such as BPMN. Such a tool feature is of particular importance to companies that have to manage large process model collections. Moreover, automatic quality assurance and customized view techniques are highly relevant for a typical industry setting of process modeling. As Rosemann [6] emphasizes, there are several practical challenges given the low level of modeling expertise that many team members of process modeling projects have. Our technique could help (novice) modelers by providing a basis for the automatic derivation of abstract models and extraction of sub-processes.

6. Conclusion

In this paper, we addressed the problem of automatically finding suitable process model names. Based on insights from theories of meaning, we defined a novel approach for proposing a name for a business process model. This approach takes the labels of activities and events of the model into account, and does not depend upon external knowledge provided by the user. We implemented the approach in a prototype tool and evaluated it with different sets of process models from practice. The results show that on average we achieve a higher semantic closeness than do the original names, and that this result is not limited to small models. In addition, we conducted a case study with a large Australian insurer. The results confirm the usefulness and appropriateness of the generated names and highlight the potential business value of our approach.

[1] `www.apromore.org`

When reading the results of the evaluation, we need to consider that companies often tend to use specific terms, and create new ones when labeling process modeling elements. This relates to two aspects: creating company-specific meaning for existing terms or creating completely new terms. In the first case, our technique is not capable of adapting to the company-specific interpretation of terms. As we rely on WordNet, any meaning not captured cannot be considered by our technique. In the second case, WordNet might not even contain the domain-specific term in question. A way to obviate this problem is to combine the use of WordNet with a company-specific thesaurus.

Our findings have implications for process model abstraction, matching of process models, and the automatic quality assurance of process models and process model repositories. In future research we aim to apply our technique for process model matching. Based on name proposals, we are able to describe parts of a process model on an aggregated level. In this way, our technique can potentially help to compare process models specified at different levels of granularity.

References

[1] T. Davenport, J. E. Short, The New Industrial Engineering: Information Technology and Business Process Redesign, Sloan Management Review 31 (4) (1990) 11–27.

[2] M. Hammer, Reengineering work: Don't automate, obliterate, Harvard Business Review 68 (4) (1990) 104–112.

[3] M. Earl, J. Sampler, J. Short, Strategies for business process reengineering: evidence from field studies, Journal of Management Information Systems 12 (1) (1995) 31–56.

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

[5] K. Altinkemer, Y. Ozcelik, Z. D. Ozdemir, Productivity and performance effects of business process reengineering: A firm-level analysis, J. of Management Information Systems 27 (4) (2011) 129–162.

[6] M. Rosemann, Potential pitfalls of process modeling: part a, Business Process Management Journal 12 (2) (2006) 249–254.

[7] I. Vessey, S. A. Conger, Learning to specify information requirements: The relationship between application and methodology, J. of Management Information Systems 10 (2) (1993) 177–202.

[8] R. Agarwal, A. P. Sinha, M. Tanniru, Cognitive fit in requirements modeling: A study of object and process methodologies, J. of Management Information Systems 13 (2) (1996) 137–162.

[9] T. Davenport, M. Beers, Managing information about processes, Journal of Management Information Systems (1995) 57–80.

[10] S. Smirnov, H. Reijers, T. Nugteren, M. Weske, Business process model abstraction: A definition, catalog, and survey, Distributed and Parallel Databases 30 (1) (2012) 63–99, to appear.

[11] D. K. W. Chiu, S. C. Cheung, S. Till, K. Karlapalem, Q. Li, E. Kafeza, Workflow View Driven Cross-Organizational Interoperability in a Web Service Environment, Information Technology and Management 5 (3–4) (2004) 221–250.

[12] R. Bobrik, M. Reichert, T. Bauer, View-based process visualization, in: BPM 2007, Vol. 4714 of LNCS, Springer, Berlin, 2007, pp. 88–95.

[13] A. Polyvyanyy, S. Smirnov, M. Weske, Process model abstraction: A slider approach, in: Proceedings of the 12th International Conference on Enterprise Distributed Object Computing (EDOC), 2008.

[14] C. C. Ekanayake, M. Dumas, L. García-Bañuelos, M. L. Rosa, A. H. M. ter Hofstede, Approximate clone detection in repositories of business process models, in: BPM, 2012, pp. 302–318.

[15] M. Dumas, L. García-Bañuelos, M. L. Rosa, R. Uba, Fast detection of exact clones in business process model repositories, Inf. Syst. 38 (4) (2013) 619–633.

[16] R. Uba, M. Dumas, L. García-Bañuelos, M. L. Rosa, Clone detection in repositories of business process models, in: BPM, 2011, pp. 248–264.

[17] J. M. Fabian Pittke, Henrik Leopold, G. Tamm, Enabling reuse of process models through the detection of similar process parts, in: 3rd International Workshop on Reuse in Business Process Management, 2012.

[18] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2013.

[19] A.-W. Scheer, Business Process Engineering: Reference Models for Industrial Enterprises, Springer-Verlag, 1994.

[20] J. Mendling, Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, Springer Publishing Company, Incorporated, 2008.

[21] Object Management Group, Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification, dtc/06-02-01, Object Management Group (February 2006).

[22] W. Aalst, A. Hofstede, YAWL: Yet Another Workflow Language, Information Systems 30 (4) (2005) 245–275.

[23] OMG, ed., Unified Modeling Language, Version 2.0, Object Management Group (2004).

[24] V. Welby, What is meaning?, Benjamins, 1983.

[25] M. Dummett, The Seas of Language, Oxford University Press, 1996.

[26] J. Evermann, Theories of meaning in schema matching: A review, Journal of Database Management (JDM) 19 (3) (2008) 55–82.

[27] J. Speaks, Theories of meaning, in: E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, summer 2011 Edition, 2011.

[28] Y. Wand, R. Weber, On the deep structure of information systems, Information Systems Journal 5 (1995) 203–223.

[29] G. Frege, On sense and reference, Philosophical Review 57 (1948/1892) 209–230.

[30] B. Russell, On denoting, Mind 56 (1905) 479–493.

[31] B. Russell, Logic and Knowledge: Essays 1901-1950, Allen and Unwin, London, 1956, Ch. Descriptions and incomplete Symbols.

[32] H. Leopold, J. Mendling, H. Reijers, On the Automatic Labeling of Process Models, in: CAiSE 2011, Vol. 6741 of LNCS, Springer, 2011, pp. 512–520.

[33] E. Rosch, Cognitive representations of semantic categories, Journal of Experimental Psychology: General 104 (3) (1975) 192–233.

[34] E. Rosch, Family resemblances: Studies in the internal structure of categories, Cognitive Psychology 7 (4) (1975) 573–605.

[35] A. Tahamtan, J. Eder, View driven federation of choreographies, in: N. T. Nguyen, R. Katarzyniak, S.-M. Chen (Eds.), Advances in Intelligent Information and Database Systems, Vol. 283 of Studies in Computational Intelligence, Springer, 2010, pp. 145–156.

[36] L. Wittgenstein, Philosophical Investigations, The German Text with an English Translation, 4th Edition, Basil Blackwell, Oxford, 2009.

[37] J. Dewey, Experience and Nature, Dover Publications, 1958.

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

[39] J. L. Austin, How to do things with words, Harvard University Press, 1975.

[40] H. P. Grice, Studies in the Way of Words, Harvard University Press, Cambridge, Massachusetts, 1989.

[41] P. Soffer, Y. Wand, Goal-driven analysis of process model validity., in: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings, Vol. 3084 of Lecture Notes in Computer Science, Springer, 2004, pp. 521–535.

[42] M. Weske, Business Process Management: Concepts, Languages, Architectures, 1st Edition, Springer Verlag, 2007.

[43] S. Kripke, Naming and necessity, Wiley-Blackwell, 1981.

[44] H. A. Reijers, S. Limam, W. M. P. van der Aalst, Product- based workflow design, J. of Management Information Systems 20 (1) (2003) 229–262.

[45] F. Gottschalk, Configurable process models, Ph.D. thesis, Eindhoven University of Technology, The Netherlands (December 2009).

[46] H. Krasner, J. Terrel, A. Linehan, P. Arnold, W. H. Ett, Lessons learned from a software process modeling system, Commun. ACM 35 (9) (1992) 91–100.

[47] 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 et al., 2000, pp. 30–49.

[48] J. Recker, M. Indulska, M. Rosemann, P. Green, An exploratory study of process modelling practice with bpmn, Tech. rep., BPM Center Report (2008).

[49] R. Dixon, Deriving verbs in english, Language Sciences 30 (1) (2008) 31–52.

[50] H. Leopold, S. Smirnov, J. Mendling, On the refactoring of activity labels in business process models (accepted for publication), Information Systems.

[51] J. Mendling, H. A. Reijers, J. Recker, Activity Labeling in Process Modeling: Empirical Insights and Recommendations, Information Systems 35 (4) (2010) 467–482.

[52] G. Decker, J. Mendling, Process instantiation, Data Knowl. Eng. 68 (9) (2009) 777–792.

[53] J. H. Lee, M. H. Kim, Y. J. Lee, Information retrieval based on conceptual distance in is-a hierarchies., Journal of Documentation 49 (2) (1993) 188–207.

[54] R. Rada, H. Mili, E. Bicknell, M. Blettner, Development and application of a metric on semantic nets, IEEE Transactions on Systems, Man, and Cybernetics 19 (1) (1989) 17–30. doi:10.1109/21.24528.
URL `http://dx.doi.org/10.1109/21.24528`

[55] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 448–453.
URL `http://dl.acm.org/citation.cfm?id=1625855.1625914`

[56] D. Lin, An information-theoretic definition of similarity, in: Proc. 15th International Conf. on Machine Learning, 1998, pp. 296–304.

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

[58] M. L. Rosa, P. Wohed, J. Mendling, A. H. M. ter Hofstede, H. A. Reijers, W. M. P. van der Aalst, Managing process model complexity via abstract syntax modifications, IEEE Trans. Industrial Informatics 7 (4) (2011) 614–629.

[59] D. Liu, M. Shen, Workflow modeling for virtual processes: an order-preserving process-view approach, Information Systems 28 (6) (2003) 505–532.

[60] A. Polyvyanyy, S. Smirnov, M. Weske, The triconnected abstraction of process models, in: BPM 2009, Vol. 5701 of LNCS, Springer, 2009, pp. 229–244.

[61] S. Smirnov, R. Dijkman, J. Mendling, M. Weske, Meronymy-based aggregation of activities in business process models, in: ER 2010, Vol. 6412 of LNCS, Springer, 2010, pp. 1–14.

[62] H. A. Reijers, J. Mendling, R. M. Dijkman, On the Usefulness of Subprocesses in Business Process Models, BPM Center Report BPM-10-03, BPMcenter.org (2010).

[63] M. Weidlich, A. Barros, J. Mendling, M. Weske, Vertical Alignment of Process Models - How Can We Get There?, in: BPMDS 2009, Vol. 29 of LNBIP, Springer, 2009, pp. 71–84.

[64] C. Di Francescomarino, A. Marchetto, P. Tonella, Cluster-based Modularization of Processes Recovered from Web

Applications, Journal of Software Maintenance and Evolution: Research and Practice.

[65] S. Smirnov, H. A. Reijers, M. Weske, A Semantic Approach for Business Process Model Abstraction, in: CAiSE 2011, Vol. 6741 of LNCS, Springer, 2011, pp. 497–511.

[66] H. A. Reijers, T. Freytag, J. Mendling, A. Eckleder, Syntax highlighting in business process models, Decision Support Systems 51 (3) (2011) 339–349.

[67] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, 2007.

[68] G. Preuner, S. Conrad, M. Schrefl, View integration of behavior in object-oriented databases, Data & Knowledge Engineering 36 (2) (2001) 153–183.

[69] A. Basu, R. W. Blanning, Synthesis and Decomposition of Processes in Organizations, Information Systems Research 14 (4) (2003) 337–355.

[70] M. La Rosa, M. Dumas, R. Uba, R. Dijkman, Business process model merging: An approach to business process consolidation, ACM Transactions on Software Engineering and Methodology 22 (2).

[71] 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.

[72] M. Ehrig, A. Koschmider, A. Oberweis, Measuring similarity between semantic business process models, in: J. Roddick, A. Hinze (Eds.), Conceptual Modelling 2007, Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007), Vol. 67, Australian Computer Science Communications, Ballarat, Victoria, Australia, 2007, pp. 71–80.

[73] D. Grigori, J. Corrales, M. Bouzeghoub, A. Gater, Ranking BPEL Processes for Service Discovery, IEEE Transactions on Services Computing 3 (3) (2010) 178–192.

[74] 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 - 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13-16, 2010. Proceedings, Vol. 6336 of Lecture Notes in Computer Science, Springer, 2010, pp. 128–144.

[75] M. Born, F. Dörr, I. Weber, User-Friendly Semantic Annotation in Business Process Modeling, in: WISE 2007 Workshops, Vol. 4832 of LNCS, Springer, 2007, pp. 260–271.

[76] C. Francescomarino, M. Rospocher, L. Serafini, P. Tonella, Semantically-Aided Business Process Modeling, in: ISWC 2009, Vol. 5823 of LNCS, Springer, Berlin, Heidelberg, 2009, pp. 114–129.

[77] 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.

[78] C. Francescomarino, P. Tonella, Supporting Ontology-Based Semantic Annotation of Business Processes with Automated Suggestions, in: BMMDS/EMMSAD 2009, Vol. 29 of LNBIP, Springer, 2009, pp. 211–223.

[79] M. Weidlich, R. Dijkman, J. Mendling, The ICoP Framework: Identification of Correspondences between Process Models, in: CAiSE 2010, Vol. 6051 of LNCS, Springer, 2010, pp. 483–498.

[80] S. Smirnov, R. Dijkman, J. Mendling, M. Weske, Meronymy-based aggregation of activities in business process models, in: Proceedings of the 29th International Conference on Conceptual Modeling, Springer, Vancouver, BC, Canada, 2010.

[81] N. Kock, J. Verville, A. Danesh-Pajou, D. DeLuca, Communication flow orientation in business process modeling and its effect on redesign success: results from a field study, Decision Support Systems 46 (2) (2009) 562–575.

[82] O. I. Lindland, G. Sindre, A. Sølvberg, Understanding quality in conceptual modeling, IEEE Software 11 (2) (1994) 42–49.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] N. Peters, M. Weidlich, Automatic Generation of Glossaries for Process Modelling Support, Enterprise Modelling and Information Systems Architectures 6 (1) (2011) 30–46.

[88] R. M. Dijkman, M. L. Rosa, H. A. Reijers, Managing large collections of business process models - current techniques and challenges, Computers in Industry 63 (2) (2012) 91–97.

[89] M. L. Rosa, H. A. Reijers, W. M. P. van der Aalst, R. M. Dijkman, J. Mendling, M. Dumas, L. García-Bañuelos, Apromore: An advanced process model repository, Expert Syst. Appl. 38 (6) (2011) 7029–7040.