Action Patterns in Business Process Model Repositories

Sergey Smirnov[a,*], Matthias Weidlich[b], Jan Mendling[c], Mathias Weske[a]

[a] *Hasso Plattner Institute, Prof.-Dr.-Helmert-Str. 2-3, D-14482 Potsdam, Germany*
[b] *Technion - Israel Institute of Technology, Technion City, 32000 Haifa, Israel*
[c] *Wirtschaftsuniversität Wien, Augasse 2-6, A-1090 Vienna, Austria*

Abstract

Business process models are extensively used in companies to document and improve business operations. In essence, there are two major challenges. The increasing number of staff with little modeling expertise involved in model design requires new concepts for quality assurance. Moreover, the huge number of process models typically maintained in a model repository impedes extraction of general process knowledge, which can be used for assistance.

This article investigates action patterns as a means to address these challenges. Action patterns capture chunks of actions often appearing together in business processes. We formalize the action pattern concept, including several types of behavioral connection, different abstraction levels, and varying action sensitivity to business objects. Our concepts are evaluated based on a prototypical implementation, which we use to extract various types of action patterns from two industrial process model collections. The results demonstrate that action patterns occurring in different application domains can be discovered.

Keywords: business process modeling, reuse in process modeling, action patterns

1. Introduction

Business process management experiences a large uptake by the industry, as more and more companies analyze and improve their business from a process-oriented perspective. Process models as formal representations of business processes facilitate many tasks in the domain of business process management. In this way, process modeling has become a daily routine of an increasing number of office workers. This development implies several challenges in terms of an efficient and effective modeling support. In particular, many staff members have low modeling competence and model only on an irregular basis [1]. For this reason, process modeling tools have to incorporate techniques to help these casual modelers to conduct their work in a productive way.

Business process modeling research has revealed several approaches to make modeling more efficient and effective. This research can be classified into two main categories. On the one hand, reference modeling aims to increase productivity based on a reuse principle: models are created for a specific domain and are meant to be customized in different application projects. On the other hand, different types of patterns describe recurring situations in a domain-independent way. The potential of both approaches is hardly reflected by current tool features. Whilst most of the pattern sets for processes and workflows are mainly used for model verification and modeling language analysis, the existing reference models are tightly coupled with their partial domain and can hardly be used in other settings. Against this background, we define a concept of *action patterns*. Action patterns build on the observation that certain *actions*, describing the work content of a textual activity label, often occur together in process models. A prominent example of such co-occurring

*Corresponding author
Email addresses: sergey.smirnov@hpi.uni-potsdam.de (Sergey Smirnov), weidlich@tx.technion.ac.il (Matthias Weidlich), jan.mendling@wu.ac.at (Jan Mendling), mathias.weske@hpi.uni-potsdam.de (Mathias Weske)

actions is a pair *accept* and *reject*. In contrast to well known workflow patterns, action patterns are closely related to the semantic content of a process model. Meanwhile, unlike reference models, action patterns are abstract enough to be applicable in various domains. As such, action patterns help to assess the quality of a process model collection and to support process modelers during the act of modeling.

The contribution of this article is a formal description of a framework for action patterns. This framework includes various types of action patterns. We define action patterns based on co-occurrence and according to the type of behavioral relationship. We also investigate the connection of patterns with certain business objects as well as different levels of abstraction to generalize action patterns. In addition, we develop an approach for the identification of patterns in existing process model collections using association rules mining. The results presented in this article extend our previous work [2, 3]. We introduced the idea of action patterns and their mining in [2] and extended these concepts by object sensitive patterns in [3]. These prior works are summarized in Sections 3 and 4.1. In this article, we integrate these dimensions into an action patterns framework that also considers the additional dimension of action generalization. Further, we present an exhaustive evaluation using two process model collections from industry. In particular, we report on findings related to the novel dimension of action generalization and investigate the potential for pattern reuse among separate model collections.

The rest of the article is structured as follows. Section 2 provides a motivating example to illustrate our approach. Section 3 formalizes the action pattern concept. Section 4 presents different ways of generalizing the action patterns concept and relating it to specific business objects. Section 5 describes the evaluation of our approach by deriving action patterns from two industrial process model collections. In Section 6 we discuss the work related to action patterns. Section 7 concludes the article.

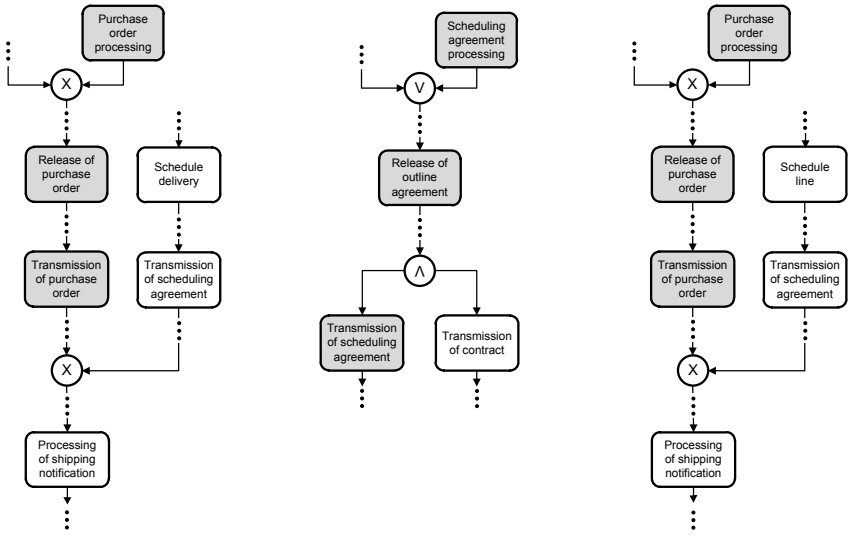
2. Background

Business process modeling efforts reach a considerable level of complexity in many companies. Commonly, a team of modelers is set up to capture the business operations of the enterprise in various process models. These models are stored and managed by the help of a central model repository as it is provided by tools like ARIS[1], ADONIS[2], or Signavio[3]. Often, such modeling initiatives result in a set of several hundreds, sometimes several thousand models [1]. Due to the large number of modelers with different levels of expertise and the sheer number of models, it is a particular challenge to keep a high level of quality. It has been shown that process model collections from practice contain errors in more than 10% of the models [4] and that a good style of labeling is hardly enforced [5]. While formal methods exist to identify these problems, there are currently no automatic techniques available to help validating process models, in particular their completeness. Therefore, there is a strong need for quality assurance and recommendation techniques to work on this issue.

Incomplete models miss parts of the process that the modeler has failed to cover. Checking completeness of a process model has to rely on the knowledge which is external to a process model. There are two important perspectives for tailoring recommendations: external knowledge can be captured in a *general or specific* way, and it can be organized in *big or small chunks*. Reference models represent a classical way of offering external knowledge for reuse in a process modeling project [6]. Reference models typically provide big chunks of specific process knowledge that can be used by a modeler as a guideline for his own model. This general idea of specific models as a reference is incorporated in the recommendation approach by Hornung et al. where similar models from a repository are shown to the modeler [7]. Big and specific models though are difficult to adapt, and modelers tend to be reluctant to delete unnecessary content [8].

The approach defined in this article is complementary to work on reference model adaptation. The idea is to identify *small and general* pieces of process knowledge, and to offer it to the modeler as a reminder for missing parts of a process model. We specifically aim to exploit the fact that big collections of process models

[1] `http://www.softwareag.com/aris/`
[2] `http://www.adonis.eu/`
[3] `http://www.signavio.com/`



(a) Procurement of materials and external services (b) Source administration (c) Third-party order processing

Figure 1: Fragments of three process models from the SAP Reference Model

already exist in a company or in terms of reference models. From these models we extract so-called action patterns. The notions of *actions* and *action patterns* are introduced in [2]. An action corresponds to the verb that describes the work content of an activity. For instance, in the activity *Purchase order processing* the action is *process*. Action patterns organize small pieces of process knowledge in terms of actions and their relations. The goal of action patterns is to define which actions often occur together in processes and which ordering constraints exist between these actions.

Let us consider an example from the SAP Reference Model [9]. Figure 1 depicts the fragments of three EPCs from this model collection. All the models describe business processes from the procurement domain. The models contain different pieces of information that we leverage using different types of action patterns. We distinguish action patterns along three dimensions that are largely orthogonal. Figure 2 illustrates the action patterns framework that is spanned by the following classification dimensions.

- *Pattern Type:* The concept of an action pattern builds on the co-occurrence of actions across different models. In Figure 1, we observe that the actions *process, release,* and *transmit* are co-occurring in all models. Furthermore, we find that there is a more specific action pattern that involves the behavioral

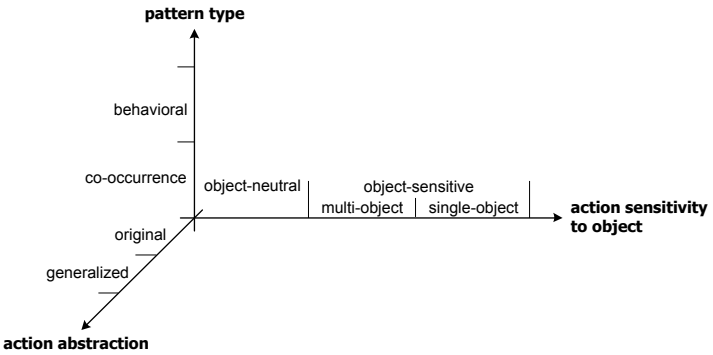


Figure 2: Action pattern framework

relation between these actions. Apparently, they are in an order relation. Hence, we distinguish co-occurrence and behavioral action patterns.

- *Action Sensitivity to Objects:* Beyond the type of a pattern, we also take into account whether the pattern is bound to a particular business object. Here, *process, release,* and *transmit* all refer to a single object: first to the *purchase order* in (a) and (c), then to the *scheduling agreement* in (b). There can be also patterns where actions are associated with different objects. According to this dimension, we distinguish object-neutral and object-sensitive action patterns, the latter relating to one or multiple objects.

- *Action Abstraction:* Patterns can be very similar in their meaning, although they are syntactically different. Assume that there would be a further process model with a pattern of *conduct, approve,* and *send*. In order to recognize the similarity with the pattern comprising the actions *process, release,* and *transmit*, we need to generalize actions. This is done based on hyponymy (is-a) relations defined in ontologies. Accordingly, we refer to these patterns as generalized action patterns.

To extract action patterns from process model collections, we rely on the following assumptions.

Assumption 1: A process model collection is large enough to extract domain knowledge. For many companies, this is not critical since hundreds of process models exist with altogether several thousand activities [1].

Assumption 2: An activity label signifies an action. Actions are either formulated as verbs or as a noun that refers to an action. Labels not referring to an action can be discarded as erroneous right from the start. In practice, there is only a small percentage of activity labels that does not meet this assumption [10].

Assumption 3: There is a mechanism interpreting an activity label as an action. Parsing an activity for its action and the corresponding business object requires natural language processing. However, the labels of process model elements are succinct pieces of text. This fact impedes the direct application of standard natural language processing methods, like part of speech tagging and discovery of grammatical structure of sentences, for activity label analysis. Against this background, the BPM community proposed a number of powerful techniques to find actions and business objects within activity labels automatically [5, 11, 12, 13]. Following the empirical findings of these research contributions this article assumes that an activity label points to one action and one object.

Assumption 4: Our approach for generalization of patterns has to rely on the availability of an ontology that defines a generalization relation for actions. Such ontologies exist. An example for process models is the MIT Process Handbook [14], which defines generalizations of activities. One may also rely on general purpose lexical databases. For instance, WordNet [15] defines hyponymy relations for terms. Further, there has been also work on the automatic discovery of such relations, see [16, 17].

In the next section, we elaborate on the concept of action patterns in detail. We first focus on a simple notion of action and the different types of patterns, co-occurrence action patterns and behavioral action patterns. Once we discuss these elementary action patterns, we extend the notion of an action pattern along the aforementioned classification.

3. Action Patterns

In general, a pattern is a concept that organizes knowledge related to "a problem which occurs over and over again in our environment, and then describes the core solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" [18]. While originally defined for architecture, this concept was adapted to software engineering in the 1990s (see [19]). In business process management, patterns have been defined, among others, for control flow [20], data flow [21], resources [22], and collaboration [23]. Also the MIT Process Handbook [14] can be related to the idea of describing a core solution to a recurring problem.

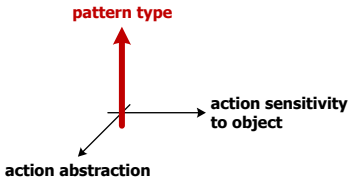


Figure 3: Action pattern classification: patterns may exploit co-occurrences only or take behavioral constraints into account

This section discusses the notion of actions patterns to meet the requirements for modeling support outlined above. In this section, we concentrate on the different types of patterns, as highlighted in Figure 3. We distinguish two types of action patterns. The first type of pattern is grounded on co-occurrences of actions. The second type additionally takes behavioral constraints between actions into account.

We proceed as follows. Section 3.1 presents our formal framework. Then, we elaborate on the notion of an action that is retrieved from a label in Section 3.2. Section 3.3 defines co-occurrence action patterns. Finally, Section 3.4 specifies behavioral action patterns based on behavioral profiles.

3.1. Formal Framework

In order to formalize the concept of an action pattern we need to introduce a number of auxiliary concepts. First, we postulate Γ to be the universal set of labels. Based thereon, we define the notion of a process model enriched with labeling information.

Definition 1 (Process Model). A tuple $PM = (A, G, F, s, e, t, l)$ is a *process model*, where:

- A is a finite nonempty set of activities;
- G is a finite set of gateways;
- $A \cap G = \emptyset$ and $N = A \cup G$ is the set of nodes;
- $s \in A$ is the start activity;
- $e \in A$ is the end activity;
- $F \subseteq (N \setminus \{e\}) \times (N \setminus \{s\})$ as the flow relation, such that $(N, F \cup \{(e, s)\})$ is a strongly connected graph;
- $t : G \mapsto \{and, xor, or\}$ is a mapping that assigns type to each gateway;
- $l : A \mapsto \Gamma$ is a mapping assigning to each activity a label.

In the remainder, we do not formalize the execution semantics of a process model, but assume an interpretation following on execution semantics of common process modeling languages. Such semantics, in particular for the OR construct, has been presented in the existing work (see [24] as an example for EPCs).

3.2. Actions as Verbs

To grasp the meaning of activities humans interpret their labels. Hence, interpretation of labels has great importance for the derivation of action patterns. At this stage, we consider the action of a label to be characterized by a verb that indicates a certain operation. This verb is a part of the label in a certain flection, or the verb stems from a noun indicating the action. We formalize this interpretation by a set of verb terms V and a verb function. The latter yields a verb for a given label.

Definition 2 (Verb Function). For a given process model $PM = (A, G, F, s, e, t, l)$, the *verb function* $v : \Gamma \mapsto V$ derives a verb from a label. As a shorthand notation, we introduce $v_a : A \mapsto \mathcal{V}$ for deriving a verb from a label of an activity $a \in A$, i.e., $v_a(a) = v(l(a))$. We also use $V_{PM} = \bigcup_{a \in A} \{v_a(a)\}$ to denote the set of all verbs of a process model PM.

As an example, one of the aforementioned process models contains the label *purchase order processing*. Applying the verb function v to this label yields the verb *process*. Further, we formalize the notion of a process model collection as follows.

Definition 3 (Process Model Collection). A tuple $C = (\mathcal{PM}, V)$ is a *process model collection* , where:

- $\mathcal{PM}$ is a nonempty finite set of process models with elements $PM_i = (A_i, G_i, F_i, s_i, e_i, t_i, l_i)$, where $i = 1, 2, \ldots, |\mathcal{PM}|$;
- $V = \bigcup_{i=1,2,\ldots,|\mathcal{PM}|} V_{PM_i}$ is the set of all actions in the model collection.

It is natural to expect that in a large collection of process models one can observe sustainable relations between actions (action patterns). Recognition of action patterns resembles uncovering patterns in large data collections. The latter problem has been addressed by data mining techniques. In particular, we are interested in association rule learning—a well established technique for discovering relations between variables in large databases. An example of an association rule in a commercial domain is a statement that if customers buy coffee and milk, they usually buy sugar as well. Association rule learning enables discovery of such statements from the analysis of basket data in supermarkets. The initial idea of association rule learning was presented by Agrawal, Imielinski, and Swami in [25]. More advanced algorithms were presented in [26].

Further, the generic formalism of association rule learning is adapted for both co-occurrence and behavioral action patterns. We introduce the set of items $\mathcal{I}$. Let us observe a collection of transactions C, where each transaction T is a set of items, i.e., $T \subseteq \mathcal{I}$. Given a set of items $X \subseteq \mathcal{I}$, we say that transaction T satisfies X, if $X \subseteq T$. An association rule in a collection C is an implication of the form $X \Rightarrow Y$, where $X \cap Y = \emptyset$ and $X, Y \subset \mathcal{I}$.

Based thereon, two elementary notions can be defined, i.e., *support* and *confidence*. A set $X \subseteq \mathcal{I}$ has support n in a collection C, if n transactions satisfy set X. We denote the support for set X with $supp(X)$. Support can be related to statistical significance. In the context of action pattern retrieval we are interested in sets that have high support. Let us require the minimum level of support for sets to be $minsup$. Then X is called a *large* set if $supp(X) \geq minsup$ (and a *small* set otherwise). An association rule $X \Rightarrow Y$ holds in transaction collection C with confidence $c = \frac{supp(X \cup Y)}{supp(X)}$, if at least c share of transactions satisfying X, satisfies Y as well. The confidence for a rule $X \Rightarrow Y$ is denoted as $conf(X \Rightarrow Y)$. A rule confidence reflects its strength. As in the case with support, we are interested in the rules with high confidence values. Hence, we introduce the minimal accepted level of confidence—$minconf$. Following [25], we claim that we are interested in the rules $X \Rightarrow Y$ for which $X \bigcup Y$ is large and the confidence is greater than user specified $minconf$.

3.3. Co-occurrence Action Patterns

The first class of action patterns is co-occurrence action patterns. As their name suggests, these patterns capture sets of actions which often co-occur together in business processes, ignoring any ordering relations between these actions. In terms of association rules learning, we interpret actions as items and process models as transactions. Hence, a model collection is a collection of transactions. We say that a process model $PM = (A, G, F, s, e, t, l)$ satisfies an action set X, if $X \subseteq V_{PM}$. A co-occurrence action pattern is defined as an association rule on the domain of actions V associated with values for minimal support and confidence.

Definition 4 (Co-occurrence Action Pattern). $CAP = (R, minsup, minconf)$ is a co-occurrence action pattern in process model collection $C = (\mathcal{PM}, V)$, where:

- R is an association rule $X \Rightarrow Y$, where $X, Y \subset V$;
- $minsup$ is the value of the required minimal support;
- $minconf$ is the value of the required minimal confidence.

From a user perspective such a pattern recommends the actions which are expected to appear in the process model given the current constellation of actions.

Mining of co-occurrence action patterns has two phases. In the first phase we seek for association rules $X \Rightarrow Y$, such that $X \bigcup Y$ is a large set. In the second phase the mined large sets are used for derivation of patterns—rules that have a high confidence level.

A search for large sets is a computationally intensive task. In this article, we set our choice on the Apriori algorithm, since it is efficient and simple [26]. In terms of large action sets this algorithm works as follows. As input, the algorithm takes the process model collection $C = (\mathcal{PM}, V)$ and the minimal support value $minsup$. For every action $v \in V$, a one element action set is constructed, $\{v\}$. Then, for each action set, the algorithm checks its support. If the support is not less than $minsup$, the set is large. The derived 1-large sets are used as the input for the next step. In the k-th step the algorithm constructs sets of size k from $k-1$ large sets and checks if they are large. The algorithm terminates, once all the large sets are found.

(a) Process model collection

Model	Actions
PM_1	allocate analyze calculate collect evaluate settle summarize
PM_2	allocate analyze asses calculate distribute entry evaluate reconcile repost settle split
PM_3	allocate analyze calculate cost settle
PM_4	allocate analyze calculate evaluate settle
PM_5	allocate analyze collect calculate distribute evaluate settle summarize
PM_6	allocate budget calculate copy define evaluate plan reconcile settle split transfer
PM_7	allocate budget calculate copy cost define plan reconcile settle split transfer

(b) Large action sets of size 1

Set	Support
{allocate}	7
{analyze}	5
{calculate}	7
{evaluate}	5
{settle}	7

(c) Large action sets of size 2

Set	Support
{allocate, analyze}	5
{allocate, calculate}	7
{allocate, evaluate}	5
{allocate, settle}	7
{analyze, calculate}	5
{analyze, settle}	5
{calculate, evaluate}	5
{calculate, settle}	7
{evaluate, settle}	5

Table 1: Derivation of large action sets in a process model collection given $minsup = 5$

For illustration, consider the model collection captured in Table 1(a). It depicts seven process models along with their sets of actions. Given a threshold of $minsup = 5$, Table 1(b) illustrates the first, and Table 1(c) the second step of the Apriori algorithm.

After large sets have been retrieved, the second phase explores each large set for rules with high confidence level. A rule $A \Rightarrow B$ is defined by two sets: antecedent (A) and consequent (B). We consider all possible partitions of a large set into two sets, one of them to become an antecedent and the other—a consequent. For each partitioning we check, if it results in a rule with a confidence level greater than $minconf$.

3.4. Behavioral Action Patterns

Co-occurrence action patterns do not provide information about *how* the missing actions have to be introduced into the process model. In other words, co-occurrence patterns do not suggest in which part of the model the activities have to be inserted. To address this question, behavioral dependencies between the actions need to be identified as well. Different behavioral models qualify as the basis to capture such dependencies. Choosing a behavioral model, again, requires balancing generality and specificity. A very specific dependency may define that two co-occurring actions, a_1 and a_2, follow each other directly, such that every occurrence of a_1 leads to an occurrence of a_2, every occurrence of a_2 is preceded by an occurrence of a_1, and no other action may occur in between. Identification of such a dependency would allow for precisely answering the question of how to insert a missing action a_2 once action a_1 is detected. On the downside, such specific patterns can be expected to be rare since it is likely that different process models allow for slight variations, e.g., action a_2 may be skipped, which violates the leads to dependency. For instance, consider the examples from the SAP Reference Model given in Section 2. Here, the shared actions are not part of an isolated region of the process models. Instead, other actions happen between, exclusive, or concurrently to the occurrences of the shared actions. Therefore, we consider a rather abstract notion of behavioral dependencies that captures relations between pairs of actions on their potential order of occurrence. As such, we leverage the information on whether the (potentially multiple) occurrences of two actions are always ordered once both actions are observed together. For the examples in Section 2, we see that the shared actions indeed have a very specific order of potential occurrence. Thus, we consider this behavioral model to be an appropriate compromise between generality and specificity of the considered behavioral dependencies. In principle, however, other models to capture behavioral dependencies may also be applied.

To enrich action patterns with behavioral information, we first present preliminaries on behavioral relations. Then, we introduce the notion of a behavioral action pattern.

3.4.1. Behavioral Relations

To capture behavioral aspects of a process on the level of pairs of activities, we apply the notion of *behavioral profiles* [27]. Although Definition 1 does not specify execution semantics, we impose syntactical requirements for the definition of all traces of a process model to define its behavioral profile. That is, the (potentially infinite) set of *traces* $\mathcal{T}_{PM}$ for a process model $PM = (A, G, F, s, e, t, l)$ is a set of lists of the form $s \cdot A^*$. That is, a list entry contains the execution order of activities. Further, we use $a \in \sigma$ with $\sigma \in \mathcal{T}_{PM}$ to denote that an activity $a \in A$ is a part of a process trace. The behavioral profile is grounded on the notion of *weak order*. Two activities of a process model are in weak order, if there exists a trace in which one node occurs after the other.

Definition 5 (Weak Order Relation). Let $PM = (A, G, F, s, e, t, l)$ be a process model, and $\mathcal{T}_{PM}$—its set of traces. The *weak order relation* $\succ_{PM} \subseteq (A \times A)$ contains all pairs (x, y), such that there is a trace $\sigma = n_1, \ldots, n_m$ in $\mathcal{T}_{PM}$ with $j \in \{1, \ldots, m-1\}$ and $j < k \leq m$ for which holds $n_j = x$ and $n_k = y$.

Depending on how two activities of a process model are related by weak order, we define three relations forming the behavioral profile.

Definition 6 (Behavioral Profile). Let $PM = (A, G, F, s, e, t, l)$ be a process model. A pair $(x, y) \in (A \times A)$ is in one of the following relations:

- *strict order relation* $\rightsquigarrow_{PM}$*, if* $x \succ_{PM} y$ *and* $y \not\succ_{PM} x$*;*
- *exclusiveness relation* $+_{PM}$*, if* $x \not\succ_{PM} y$ *and* $y \not\succ_{PM} x$*;*
- *interleaving order relation* $||_{PM}$*, if* $x \succ_{PM} y$ *and* $y \succ_{PM} x$*.*

The set of all three relations is the *behavioral profile* of PM.

We illustrate the behavioral profile by means of the model in Figure 4. For instance, it holds *(Template allocation)* $\rightsquigarrow$ *(Overhead calculation)*. There exists a trace in which the template allocation occurs before the overhead calculation. In contrast, there does not exist a trace in which both functions occur in the reversed order, i.e., the overhead calculation is never followed by the template allocation. With $\rightsquigarrow^{-1}_{PM}$ as the inverse relation for $\rightsquigarrow_{PM}$, *(Overhead calculation)* $\rightsquigarrow^{-1}_{PM}$ *(Template allocation)* also holds. An example for the exclusiveness relation would be functions *(Schedule for revaluation)* and *(Overhead calculation)*. Both never occur together in a trace of the model. It is worth to mention that $\rightsquigarrow_{PM}$,$\rightsquigarrow^{-1}_{PM}$,$+_{PM}$, and $||_{PM}$ partition the Cartesian product of activities $A \times A$ for a process model $PM = (A, G, F, s, e, t, l)$ [27].

3.4.2. The Concept of Behavioral Action Patterns

We introduce behavioral action patterns as a mechanism enabling suggestions on how the missing actions should be introduced in an existing process model. Such patterns provide more information to the user than co-occurrence action patterns. However, we perceive behavioral patterns not as a mechanism replacing co-occurrence patterns, but rather as a complimentary mechanism: while co-occurrence action patterns suggest which actions are missing, behavioral action patterns hints on action relations. Assume a user designs a process model containing actions *allocate* and *calculate*; co-occurrence action pattern $\{allocate, calculate\} \Rightarrow \{settle\}$ is available (see Figure 4). This pattern suggests to add action *settle* in the process model. Then, we can look up a suitable behavioral action pattern describing relations between these three actions. Behavioral action pattern $\{allocate \rightsquigarrow calculate\} \Rightarrow \{allocate \rightsquigarrow settle,\ calculate \rightsquigarrow settle\}$ provides a desired recommendation.

To formalize the concept of relations between actions, we propose to adapt the behavioral relations between activities introduced earlier. We say that actions v_1 and v_2 are in relation R in a process model $PM = (A, G, F, s, e, t, l)$, if there are two activities $a, b \in A$, such that $(a, b) \in R \wedge v_a(a) = v_1 \wedge v_a(b) = v_2$. Within one process model a pair of actions (v_1, v_2) may be in more than one relation. This holds if there are several activities that signify action v_1, or action v_2, or both actions.

Definition 7 (Behavioral Action Pattern). $BAP = (R, minsup, minconf)$ is a *behavioral action pattern* in process model collection $C = (\mathcal{PM}, V)$, where:

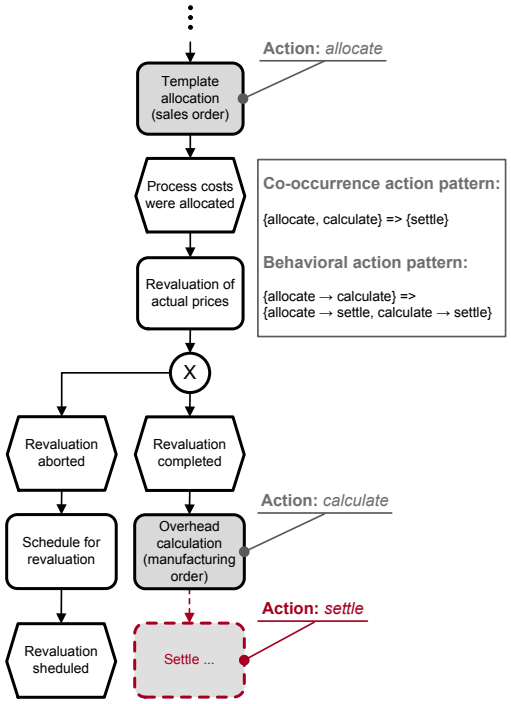


Figure 4: Exemplary suggestion based on action patterns

- R is a rule $X \Rightarrow Y$, where $X, Y \subset V \times \{\rightsquigarrow, \rightsquigarrow^{-1}, +, ||\} \times V$, i.e., X and Y constitute of pairs of actions for which behavioral relations are specified;
- *minsup* is the value of the required minimal support;
- *minconf* is the value of the required minimal confidence.

Mining of behavioral action patterns resembles the approach introduced for co-occurrence action patterns. In the first phase we seek for large action sets. In the second phase we inspect the relations between the actions of each large set. In terms of association rules derivation, action relations are treated as items, while large action sets are interpreted as collections. Once *minsup* and *minconf* values are provided, we can derive behavioral action patterns.

4. Advanced Action Patterns

Elementary action patterns as introduced in the previous section can be seen as being very specific towards the action, but unspecific towards the action context. On the one hand, we considered only equal actions and neglected that certain actions may be very close to each other. On the other hand, actions have been treated independently of their context. That is, we did not distinguish two actions that are applied to different objects.

Taking the presented model of action patterns as a basis, this section extends the notion of action patterns in these directions. First, Section 4.1 introduces different types of object-sensitive action patterns. Then, we propose generalized action patterns in Section 4.2.

4.1. Object-Sensitive Action Patterns

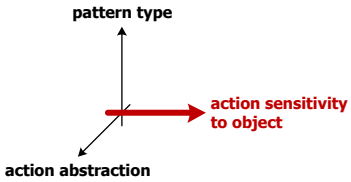


Figure 5: Exploring actions with respect to object sensitivity

The action patterns introduced so far are independent of the objects on which the actions are performed. We refer to such action patterns as *object-neutral action patterns*. *Object-sensitive action patterns* consider the context of an action, i.e., the object to which an action is applied. This dimension of our action patterns framework is highlighted in Figure 5. To introduce object-sensitive action patterns, we first elaborate on the limitations of interpreting an action as being a verb only. Subsequently, we introduce two notions of action sets that underlie object-sensitive action patterns.

4.1.1. Object-Sensitive Actions

Arguably, two actions such as *accept* and *reject* often occur together and are executed exclusively. However, the two actions provide no information about the business objects to which they are applied. Refining the action notion by taking into consideration the respective object helps to derive more detailed and precise patterns that provision additional information to the user. For instance, in the example in Figure 1, we observe that all models contain the actions *process* and *release*. However, the actions relate to different objects even within one model, i.e., a *purchase order*, a *shipping notification*, and a *scheduling agreement*. While this does not impact on the co-occurrence of these actions at least in our example, we observe that the three models show different behavioral relations between these actions. In Figure 1(a) and Figure 1(c) the action *release* (of a purchase order) can be preceded and followed by the action *process* (of a purchase order, or of a shipping notification). In contrast, the model in Figure 1(b) shows a strict order of actions: *process* is followed by *release*.

We see that object-neutral action patterns aim at deriving patterns at a coarse-grained level. Certain patterns, however, may solely be distinguished once a more fine-grained approach is taken. Such fine-grained patterns consider the combination of a verb and the respective object as the underlying notion of an action. We refer to these patterns as *object-sensitive action patterns*.

The definition of object-sensitive action patterns leads to a different domain for an action pattern, but does not impact on the action pattern types, i.e., co-occurrence and behavioral action patterns. Hence, object-sensitive patterns are mined in exactly the same way as object-neutral patterns, so that the approach introduced in Section 3 is used. Therefore, we focus on the definition of the domain for object-sensitive action patterns in this section.

Given a process model, we assume means to extract business objects from activity labels. Similar to the set of verb terms V and the verb function, we formalize this assumption by a set of object terms B and an object function.

Definition 8 (Business Object Function). For a given process model $PM = (A, G, F, s, e, t, l)$, the *object function* $b : \Gamma \mapsto B$ derives a business object from an activity label. As a shorthand notation, we introduce $b_a : A \mapsto B$ for deriving a business object from a label of an activity $a \in A$, i.e., $b_a(a) = b(l(a))$. We also use $B_{PM} = \bigcup_{a \in A} \{b_a(a)\}$ to denote the set of all business objects of a process model.

As an example, consider the label *Schedule delivery*. Then, the verb function yields *v(Schedule delivery) = schedule* and the business object function results in *b(Schedule delivery) = delivery*.

As for the case of verb terms, the notion of a set of object terms is lifted from a process model to the level of a process model collection. Hence, a process model collection is formalized a tuple $C = (\mathcal{PM}, V, B)$ with $B = \bigcup_{i=1,2,\ldots,|\mathcal{PM}|} B_{PM_i}$ as the set of all business objects in the model collection. Apparently, object-neutral action patterns are build solely from the set of verb terms V. That is, verbs are the domain for these patterns and represent the items in the sense of association rule learning, see Section 3.

4.1.2. Multi-Object Action Patterns

The first kind of object-sensitive action patterns builds on the notion of actions in the sense of an operation expressed by a verb and applied to a business object. Actions become tuples of verbs and business objects, and every such tuple is considered as an item in the sense of association rule learning.

Definition 9 (Multi-Object Action Set). Let $C = (\mathcal{PM}, V, B)$ a process model collection, the *multi-object action set* $\mathcal{A}_O \subseteq V \times B$ contains all pairs of verbs and objects (x, y), such that there exists a process model $PM = (A, G, F, s, e, t, l) \in \mathcal{PM}$ and an activity $a \in A$ with $v_a(a) = x$ and $b_a(a) = y$.

We speak of multi-object action sets, as the actions can relate to different business objects. For instance, *(process, purchase order)*, *(release, purchase order)*, and *(transmit, scheduling agreement)* would be actions derived from the exemplary process models in Figure 1 that can be part of a multi-object action pattern. An example for a co-occurrence action pattern is {*(process, purchase order), (release, purchase order)*} $\Rightarrow$ {*(transmit, scheduling agreement)*}, i.e., the observation of the actions *(process, purchase order)* and *(release, purchase order)* suggests that the action *(transmit, scheduling agreement)* should be observed as well. Similarly, behavioral action patterns can be specified, e.g., if actions *(process, purchase order)* and *(release, purchase order)* are observed in a strict order, action *(transmit, scheduling agreement)* should be observed exclusively to both.

Multi-object action patterns are more fine-grained than object-neutral action patterns. In contrast to object-neutral patterns, multi-object action patterns provide additional information about the objects to which the actions are applied. On the other hand, identification of object-neutral action patterns should be prioritized as several multi-object action patterns together might represent an object-neutral action pattern.

4.1.3. Single-Object Action Patterns

Single-object action patterns are composed of actions where verbs are applied to a *single* business object. Again, tuples of verbs and business objects are the items in the sense of association rule learning. However, in contrast to multi-object action patterns discussed in the previous section, all actions of a pattern relate to one dedicated business object. Focusing on the actions applied to a particular business object, single-object action patterns move further along the dimension of specificity. In this context, we need the following definition of an action.

Definition 10 (Single-Object Action Set). Let $C = (\mathcal{PM}, V, B)$ a process model collection. For each object $o \in B$, the *single-object action set* $\mathcal{A}^o_O \subseteq V \times B$ contains all pairs (x, o), such that there exists a process model $PM = (A, G, F, s, e, t, l) \in \mathcal{PM}$ and an activity $a \in A$ with $v_a(a) = x$ and $b_a(a) = o$.

Regarding the models in Figure 1, again, the actions *(process, purchase order)* and *(transmit, scheduling agreement)* are derived. However, these actions refer to different business objects and, therefore, cannot occur together in a single-object action pattern. Still, we see that the models in Figure 1 contain various actions that refer to a business object, the *purchase order* object. All these actions might be used as building blocks for single-object action patterns.

4.2. Generalized Action Patterns

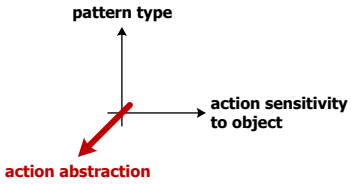


Figure 6: Exploring action patterns with generalized actions

Once we showed how action patterns may be concretized by considering objects, we discuss a generalization for the notion of an action. Again, this dimension of the action patterns framework is independent of the dimensions discussed so far. Figure 6 highlights the dimension of action generalization. First, we motivate this kind of generalization by discussing vocabulary heterogeneity often observed in modeling projects. Then, we provide an operationalization of action generalization, which allows the retrieval of generalized action patterns.

4.2.1. Generalized Actions

One of the challenges in activity labeling quality is the use of terms that have a similar meaning or are even synonyms. The impact of a vocabulary enriched with such terms is twofold. If the use of terms with slight semantic differences is intended, a large vocabulary allows the modeler to achieve the required precision

expressing the nuances of actions. Upon the other hand, if the interchange of such terms is uncontrolled, the model labeling quality decreases.

The usage of terms that are semantically close or synonymous has a negative impact on our framework. It impedes pattern discovery, since it lowers the support and confidence values observed for action patterns. Consider an example, in which ten models contain a sequence of activities *Change order* and *File order* and 5 models contain a sequence of activities *Update order* and *Archive order*. The two activity sequences have very close semantics, if not the same. Meanwhile, a direct search for action patterns ignores this semantic proximity.

To control the influence of semantically close terms on action pattern discovery, we rely on a notion of action generalization. In particular, we exploit a hyponymy relation of verbs. The hyponymy relation, also known as is-a relation or specialization relation, organizes verbs into a hierarchy, in which the most generic verb appears at the top. As an example, Figure 7 depicts a hyponymy tree in which node v_r represents the most generic verb, i.e., *act*. The deeper we advance in the hierarchy, the more specific verbs we observe. Thereby, the leaves of the hyponymy tree correspond to most specific verbs. Notice that every verb hyponymy hierarchy trivially defines a verb vocabulary, i.e., a set of verb terms. To restrict the set of verb terms, we consider a horizontal cut in the respective hierarchy. Such a cut is specified by a distance from the hierarchy root. The closer the cut is to the hierarchy root, the more generic are the verbs in the restricted set of verb terms.

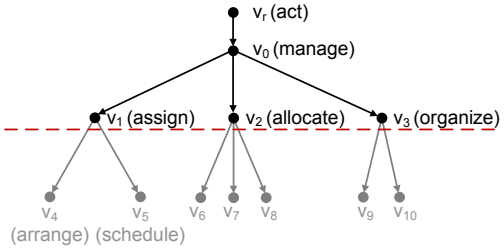


Figure 7: An example hyponymy tree

We formalize the assumption of a hyponymy relation for a set of verb terms V by a hyponymy function. The hyponymy function defines the generalization of each verb except the verb corresponding to the root node of a hyponymy tree. Note that the set of verbs is not necessarily related to a collection of process models since this set along with the hyponymy function represent external knowledge.

Definition 11 (Hyponymy Function). For a set of verbs V_h, the *hyponymy function* $h : (V_h \backslash \{v_r\}) \mapsto V_h$, $v_r \in V_h$, associates to a verb its generalized verb, such that it creates a tree hierarchy, i.e., there is exactly one verb that is not part of the domain of h, the root verb v_r, and the transitive closure of h is acyclic. The $n-$restricted verb set $V_h^n \subseteq V_h$, $n \in \mathbb{N}$, contains all verbs $v \in V_h$ such that there exists a sequence of verbs $v_1, \ldots, v_m$, $m \in \mathbb{N}$, $m \leq n$ with $v_i \in V_h$ for $i \in \mathbb{N}$, $1 \leq i \leq m$, $h(v_j) = v_{j+1}$ for $j \in \mathbb{N}$, $1 \leq j < m$, and $v_1 = v$ and $v_m = v_r$.

Taking up the example illustrated in Figure 7, we obtain *h(arrange)=assign* and *h(assign)=manage*. Further, Figure 7 visualizes a cut with a dashed line. It results in a restricted set of verb terms that includes the verbs *act, manage, assign, allocate*, and *organize*.

4.2.2. Stepwise Generalization of Action Patterns

Once a restricted hyponymy hierarchy is available, we perform the generalization for the actions of a process model collection. We use the restricted hyponymy hierarchy to generalize the verb of an action (the action may be a pair of a verb and an object). First, a verb is located in the original verb hyponymy hierarchy. Second, we traverse through this hierarchy towards the root until we find the first verb that belongs to the restricted hierarchy. The found verb is then taken to build the generalized action.

The definition of generalized action patterns changes the domain for action patterns. As for the case of object-sensitive patterns, however, it neither impacts on the action pattern types, i.e., co-occurrence and behavioral action patterns, nor on the approach to mine action patterns. Further, the generalization of verbs is independent of the extent to which objects are considered. Hence, we formally define the generalization of a set of actions only for the case of object-neutral actions. Again, generalization is parameterized by the depth of the cut used to restrict a hyponymy hierarchy.

Definition 12 (Generalized Action Set). Let $C = (\mathcal{PM}, V)$ a process model collection and V_h a set of verb terms for which h is a hyponymy function. The *n-generalized action set* $\mathcal{A}_G^n$ contains all verbs that are in the $n-$restricted set of verb terms of the hyponymy hierarchy, i.e., $\mathcal{A}_G^n = V_h^n$.

For the example illustrated in Figure 7, we would derive action patterns that are build from the restricted set of verb terms, *act, manage, assign, allocate*, and *organize*. Since action generalization is realized by the navigation through the action hyponymy hierarchy, the role of this hierarchy can not be underestimated. The nodes of the tree (actions), as well as the tree structure (hyponymy relation), specify how generalization is implemented and directly impact the resulting action patterns. Hence, the designer should carefully select the action hierarchy. While in some cases the already available hierarchies, like the MIT Process Handbook, suffice, there might be a need for the creation of a customized hierarchy. Although the latter case requires more effort, it allows the modeler to realize generalization more precisely addressing the characteristics of the process model collection at hand.

Apparently, the goal of action generalization is to increase the support and confidence values for action patterns. However, one can foresee that extensive generalization has also a negative effect, since it leads to information loss. As soon as several actions are stemmed to one, the details of the original labeling are lost. Thereby, it is important to find a trade-off between information loss and reliable discovery of action patterns. The question of what is an appropriate balance cannot be answered in the general case, but has to be addressed for a specific use case and model collection. Still, the parameterization of the generalization allows to adapt the approach towards a dedicated setting. In this vein, we distinguish two important action patterns use cases: the reuse within one process model collection and inter-collection reuse. If action patterns are reused within one process model collection, moderate generalization may suffice. It allows preserving the intended specificity of vocabulary, yet discovering the patterns more reliably. At the same time, if action patterns are intended for a reuse between various model collections, stronger generalization might be more appropriate. Since the vocabularies of different collections may vary heavily, extensive generalization facilitates pattern discovery at the expense of large information loss. We further investigate this trade-off with our experimental evaluation outlined in Section 5.

5. Experimental Evaluation

To validate the action patterns framework, we have conducted a series of experiments. The goals of the experiments were:

1. To learn which support and confidence values are encountered in practice.
2. To evaluate the potential of action patterns reuse among independent model collections.

This section is structured according to the two research questions. Before we turn to these questions, we outline the experiment setting.

5.1. Experiment Setting

The experiment studies two process model collections: the SAP Reference Model [9] and the Telecom collection. The SAP Reference Model has been used in several works on process model analysis [5, 2, 24]. The collection captures business processes that are supported by the SAP R/3 software in its version from the year 2000. It is organized in 29 functional branches of an enterprise, like sales or accounting, covered by the SAP software. The SAP Reference Model includes 604 Event-driven Process Chains (EPCs). All of these models have been considered in the phase of the experiment where co-occurrence patterns are derived. The experiment phase inspecting behavioral action patterns, studied 421 model. The decrease in the model number is resulted by the exclusion of models with ambiguous instantiation semantics, see [28], or behavioral anomalies, see [24]. Exclusion of these models is motivated not only technically, but also by the use case. Since action patterns aim at increasing the reuse of the knowledge available in process models, it is important that the input for the action patterns derivation is of high quality. Even though the question of how to assess the quality of a process model collection is a challenging research question, ambiguous instantiation semantics and behavioral anomalies are a clear indicator for bad model quality. Therefore, externalizing the knowledge encoded by these models for reuse is inadequate. Instead, the problems of these models should be

minconf \ *minsup*	2	3	4	5	6	7	8	...	23
0.50	522396	7395	2353	680	563	41	29	...	1
0.55	511373	6979	2247	665	550	34	23	...	1
0.60	510517	6123	2089	610	504	33	22	...	1
0.65	510498	6104	2070	591	497	26	16	...	1
0.70	484061	5569	1535	563	469	20	12	...	1
0.75	483415	4923	1477	505	421	19	11	...	1
0.80	483176	4684	1238	501	417	15	10	...	1
0.85	483135	4643	1197	460	417	15	10	...	1
0.90	483095	4603	1157	420	377	7	3	...	0
0.95	483093	4601	1155	418	375	5	1	...	0

Table 2: Dependency of co-occurrence pattern number in the SAP Reference Model on *minsup* and *minconf* values

corrected before they may be considered as part of the knowledge basis. To meet the assumptions on means to interpret labels, we relied on manual tagging. Overall, we discovered 215 distinct verbs, i.e., object-neutral actions, in the labels of the SAP Reference Model.

The Telecom collection describes the business processes of a large telecommunication enterprise as of year 2010. It contains 486 business process models that have been created for documentation purposes. The models span various company divisions and cover several projects. Again, we had to exclude various models due to syntactic or semantic issues for the experiment on behavioral action patterns. The latter explored 253 process models. Again, interpretation of labels was done manually. In the Telecom collection, we identified 323 distinct verbs, i.e., object-neutral actions.

For the action generalization, we relied on the MIT Process Handbook [14]. The MIT Process Handbook describes business processes elicited by researchers in the interviews with business process experts. It spans several business domains, like sales, distribution, and production. The handbook describes about 5 000 activities and specifies hyponymy and meronymy relations between them. For action generalization, we relied on the activity hyponymy hierarchy. Each verb observed in a collection is generalized to its parent verb in the hierarchy. As for the configuration of the generalization by the depth of the restriction of the hyponymy hierarchy, we consider two generalization scenarios: a verb is generalized to its parent that belongs to 1) the third level in the hyponymy hierarchy or 2) the second level in the hierarchy.

5.2. Understanding Action Patterns Support and Confidence

We begin exploring object-neutral co-occurrence action patterns. The first question to be answered is which values of support and confidence indicate relevant patterns. While higher values indicate that a pattern is more reliable, we aim to understand which values can be expected. In the SAP Reference Model, the support value for all action sets is under 24, which is rather low given the fact that some actions appear several hundred times [29]. As the minimally acceptable confidence level is hard to predict, we conducted a set of experiments varying the level of support from 2 till the level when no pattern is observed. We also changed the level of confidence from 0.5 to 0.95. Table 2 summarizes the results of these experiments. It shows that there is almost half a million patterns with support of 2, 17 patterns with support of 9, and not a single pattern has support of 24. To illustrate how the derived patterns look like, we zoom into one cell of the table and list the patterns with $minsup = 7$ and $minconf = 0.95$:

1. {pick} ⇒ {process}
2. {level} ⇒ {evaluate}
3. {permit} ⇒ {process}
4. {archive, enter} ⇒ {process}
5. {allocate, calculate} ⇒ {settle}

Behavioral action patterns originate from the inspection of behavioral constraints between actions in large action sets. Hence, derivation of behavioral patterns is possible only after *minsup* for action sets is given. Table 3(a) provides an example of behavioral relations for the actions *allocate, calculate,* and *settle* in four different process models. Table 3(b) shows the number of behavioral action patterns that can be derived for this set depending on the *minsup* and *minconf* values for the relations. A concrete example of a behavioral

(a) Behavioral relations for pairs of actions

Model \ Action pair	(A, C)	(A, S)	(C, S)
PM_1	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$
PM_2	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$
PM_3	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$
PM_4	$\rightsquigarrow$	$\rightsquigarrow$	$\rightsquigarrow$

(b) Dependency of behavioral action patterns number on *minsup* and *minconf* values

minconf \ *minsup*	2	3	4	5
0.50	12	12	12	0
0.60	12	12	12	0
0.70	12	12	12	0
0.80	12	12	12	0
0.90	12	12	12	0

Table 3: Behavioral properties observed for action set {allocate (A), calculate (C), settle (S)}

action pattern which can be derived from Table 3(a) is the pattern $\{allocate \rightsquigarrow calculate\} \Rightarrow \{allocate \rightsquigarrow settle,\ calculate \rightsquigarrow settle\}$. It describes that the three actions are sequentially ordered. They should appear in the process model such that we first *allocate*, then *calculate*, and finally *settle*. This is a standard sequence of activities for financial assets.

To see whether the support values obtained for the SAP Reference Model are comparable to those obtained for the Telecom collection, we also derive the respective patterns for the latter collection. Table 4(a) compares the amount of discovered object-neutral co-occurrence action patterns for the SAP Reference Model and the Telecom collection. Notice that the number of object-neutral co-occurrence action patterns in the Telecom collection with the $minsup = 2$ is absent: the number of patterns is so large that it was not feasible to compute it within the experiment. Table 4(a) illustrates that we observe more action patters in the Telecom collection, even though it comprises less models than the SAP Reference Model. Apparently, for both model collections the number of patterns decreases with the growth of support and confidence values (horizontal and vertical directions in the table, respectively). It is remarkable though, that both collections comprise a lot of action patterns with a very high confidence level for the medium support levels of four to eight. These patterns with confidence levels of 0.8 or 0.9 are of high practical value, since missing actions can be detected with a low error rate.

As a next step, we investigate the influence of action generalization for the discovered object-neutral co-occurrence action patterns. Generalization stems several semantically close actions to one action. Subsequently, multiple action patterns merge into one pattern. Taking the hyponymy tree of the MIT Process Handbook as an example, the following three patterns yield one pattern {assign} ⇒ {modify} once generalization is applied.

1. {arrange} ⇒ {modify}
2. {schedule} ⇒ {modify}
3. {assign} ⇒ {modify}

As a consequence of the generalization, we observe an increase in support values. Table 4(b) depicts the results obtained for a restriction of the hyponymy hierarchy to the third level. These abstract action patterns are observed more often than the original ones. In the SAP Reference Model, we observe support values of up to 34, and for the Telecom collection even up to 218. On the down side, however, the discovered patterns are not as distinguishing as before. In particular for the case of the SAP Reference Model, we only observe significantly less patterns with high confidence values. Table 4(c) confirms these observations by depicting the results obtained for an even stronger generalization. Here, the hyponymy hierarchy is restricted to the second level, so that the verbs that form the object-neutral actions are more general than in the previous case.

So far, all the discovered action patterns have been object-neutral. In Table 5, we focus on the action sensitivity to objects. Tables 5(a)–(c) consider patterns built from single-object action sets. Besides using the original verbs to construct actions, we also apply the two steps of verb generalization. Tables 5(d)–(f) capture the results for multi-object action sets.

The number of patterns for multi-object action sets is the highest among the considered pattern types. This can be explained by the fact that one action pattern consisting of object-neutral actions is "split" into several patterns capturing object-sensitive actions. Consider an example object-neutral action set {*transmit,*

(a) Object-neutral actions without generalization

minconf \ *minsup*	2	3	4	5	6	7	8	23	2	4	8	16	32	36
0.50	522396	7395	2353	680	563	41	29	1	–	9218	1107	12	1	0
0.60	510517	6123	2089	610	504	33	22	1	–	9218	1107	12	1	0
0.70	484061	5569	1535	563	469	20	12	1	–	5439	521	2	0	0
0.80	483176	4684	1238	501	417	15	10	1	–	3631	297	2	0	0
0.90	483095	4603	1157	420	377	7	3	0	–	2993	121	0	0	0

(b) Object-neutral actions generalized to the third level in the action hyponymy hierarchy

minconf \ *minsup*	2	4	6	8	10	12	14	16	34	2	4	8	64	128	218
0.50	21981	621	144	61	34	28	19	17	4	101966	7773	2040	20	2	2
0.60	19627	466	97	36	16	14	10	9	1	85287	5766	1531	19	2	2
0.70	16376	336	73	24	9	8	5	4	1	52055	4253	1007	15	1	1
0.80	16031	238	43	11	3	3	2	2	1	48416	3400	693	9	0	0
0.90	15998	205	29	4	1	1	0	0	0	46953	1937	404	1	0	0

(c) Object-neutral actions generalized to the second level in the action hyponymy hierarchy

minconf \ *minsup*	2	4	8	16	32	64	128	179	2	4	8	64	128	218
0.50	265	116	59	28	18	7	3	1	1292	669	617	71	22	2
0.60	208	85	49	23	13	6	3	0	1098	547	504	63	19	2
0.70	169	66	33	13	6	2	0	0	897	436	394	41	11	2
0.80	135	41	19	6	2	0	0	0	737	330	297	25	6	0
0.90	120	26	10	3	1	0	0	0	581	174	159	6	0	0

Table 4: Dependency of co-occurrence pattern number in the SAP Reference Model and Telecom collection on *minsup* and *minconf* values (SAP Reference Model on the left)

process, release}. In the case of object-sensitive actions this set is split into 22 action sets, including:

1. {*transmit order, process order, release order*}
2. {*transmit agreement, process notification, release order*}
3. {*transmit order, transmit notification, process order, release order*}

These examples cover different types of object-sensitive actions. The first set illustrates a single-object action set. The second set is a multi-object action set performed on objects *order*, *agreement*, and *notification*. Finally, the third set shows that there are multi-object action sets, where one verb is performed on more than one object, e.g., *transmit order* and *transmit notification*. Due to this "split" of actions, the number of patterns describing object-sensitive actions can be greater than the number of patterns for object-neutral actions. Meanwhile, the support value of each individual pattern capturing object-sensitive actions is lower than the support for a pattern for object-neutral actions: the support of one generic pattern is distributed among several more specific patterns. Table 5 shows that the more specific are the patterns, the bigger is the share of patterns with high confidence.

Table 5 also illustrates the effect of generalization for object-sensitive action patterns: the number of distinct action patterns abates by the magnitude of orders. At the same time, we do not observe an increase in the support values. The latter phenomenon can be attributed to the observation that object-sensitive patterns take objects into account. In this setting, the number of models where a particular object appears limits the support value for an object-sensitive action pattern.

Finally, we focus on the negative effect of action generalization in more detail. Generalization implies information loss. As mentioned above, generalization leads to the combination of multiple action patterns into one pattern. For illustration purposes, consider the following set of action patterns mined from the SAP Reference Model (*minsup = 8* and *minconf = 0.95*) using action generalization to the second level of the action hyponymy hierarchy in the MIT Process Handbook.

1. {preserve} ⇒ {modify}
2. {act, preserve} ⇒ {modify}

(a) Single-object patterns where actions are not generalized

minconf \ *minsup*	2	3	4	5	6	7	10	2	3	4	5	6	7	8
0.50	976	408	119	67	53	10	1	2418	105	25	8	3	3	1
0.60	937	397	108	63	53	10	1	2252	90	18	6	1	1	0
0.70	858	364	87	56	46	9	0	1772	80	15	5	1	1	0
0.80	830	336	83	52	43	7	0	1753	61	13	3	0	0	0
0.90	805	311	58	37	29	6	0	1749	57	9	2	0	0	0

(b) Single-object patterns where actions are generalized to the third level in the action hyponymy hierarchy

minconf \ *minsup*	2	3	4	5	6	7	10	2	3	4	5	6	7	8
0.50	321	131	57	34	26	6	1	375	47	14	5	2	2	1
0.60	311	125	51	32	26	6	1	362	44	13	4	2	2	0
0.70	304	124	50	31	25	6	1	291	39	10	3	1	1	0
0.80	298	118	49	30	24	5	1	285	33	9	2	0	0	0
0.90	288	108	39	27	21	4	0	283	30	6	1	0	0	0

(c) Single-object patterns where actions are generalized to the second level in the action hyponymy hierarchy

minconf \ *minsup*	2	3	4	5	6	7	10	2	3	4	5	6	7	8
0.50	253	101	43	26	22	6	1	280	45	14	5	2	2	1
0.60	250	99	41	26	22	6	1	269	42	13	4	1	1	0
0.70	245	99	41	26	22	6	1	228	37	10	3	1	1	0
0.80	241	94	41	26	22	5	1	223	31	9	2	0	0	0
0.90	235	88	34	24	19	4	0	221	28	6	1	0	0	0

(d) Multi-object patterns where actions are not generalized

minconf \ *minsup*	2	3	4	5	6	7	10	2	3	4	5	6	7	9
0.50	–	32072	21019	6171	6010	20	1	–	57198	86	25	5	4	1
0.60	–	30991	20376	6157	6010	20	1	–	57092	68	19	3	2	1
0.70	–	30991	14130	6013	5870	19	1	–	56466	51	16	2	2	1
0.80	–	20996	14126	6009	5867	17	1	–	56258	48	13	1	1	1
0.90	–	20301	13431	5709	5589	10	0	–	56250	40	11	1	1	1

(e) Multi-object patterns where actions are generalized to the third level in the action hyponymy hierarchy

minconf \ *minsup*	2	3	4	5	6	7	10	2	3	4	5	6	7	9
0.50	–	3193	1933	675	618	12	1	–	2340	46	14	4	3	1
0.60	–	3166	1916	673	618	12	1	–	2310	41	11	3	2	1
0.70	–	2957	1718	669	615	12	1	–	2286	26	10	2	2	1
0.80	–	2925	1717	668	614	11	1	–	2244	25	8	1	1	1
0.90	–	2865	1657	647	604	6	0	–	2241	22	7	1	1	1

(f) Multi-object patterns where actions are generalized to the second level in the action hyponymy hierarchy

minconf \ *minsup*	2	3	4	5	6	7	10	2	3	4	5	6	7	9
0.50	–	1888	1204	448	404	12	1	–	2335	46	14	4	3	1
0.60	–	1869	1195	448	404	12	1	–	2305	41	11	3	2	1
0.70	–	1729	1064	445	402	12	1	–	2281	26	8	2	2	1
0.80	–	1710	1064	445	402	11	1	–	2239	25	7	1	1	1
0.90	–	1679	1032	432	395	6	0	–	2236	22	6	1	1	1

Table 5: Dependency of co-occurrence object-sensitive action pattern number in the SAP Reference Model and Telecom collection on *minsup* and *minconf* values (SAP Reference Model on the left)

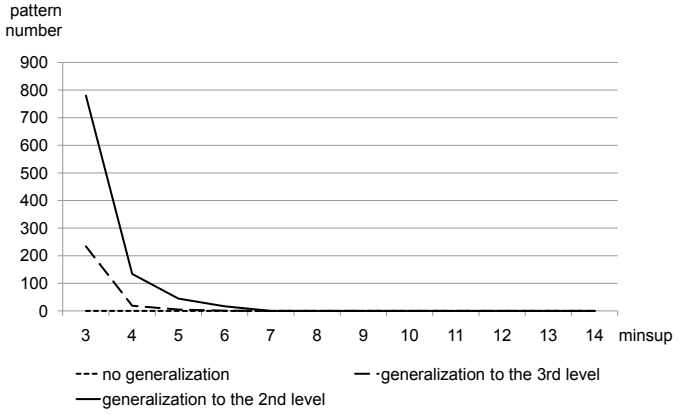


Figure 8: The number of shared co-occurrence action patterns in the SAP Reference Model and Telecom collection, $minconf = 0.9$

3. {act, decide, destroy} ⇒ {modify}
4. {decide, destroy, modify} ⇒ {act}

In comparison to the patterns where actions are not generalized, these patterns operate with a vocabulary of actions restricted by half. Thus, generalization leads to a loss of semantic precision of the original terms used in the process model collection. As a consequence, action patterns operating with generalized actions bring less knowledge than patterns with original actions.

Against this background, we conclude that the user has to balance action generalization and action patterns support according to the task at hand. For instance, if the reuse of action patterns is limited to one process model collection, actions without generalization or actions with low level of generalization are appropriate. This design decision would help to preserve the collection-specific vocabulary, yet pattern support values can be quite high. On the other hand, inter-collection action pattern reuse stimulates higher information loss and implies stronger action generalization.

5.3. Action Patterns Reuse

Action patterns aim at externalizing the knowledge hidden in process model collections. With that respect, we assess the quality of action patterns. One quality criterion is the degree of reuse for the discovered knowledge. To evaluate this aspect, we consider two independently created process model collections and estimate the number of shared action patterns, i.e., action patterns observed both in the SAP Reference Model and the Telecom collection.

Figure 8 reflects the number of shared co-occurrence action patterns for a *minconf* value of 0.9. The number of patterns operating with original actions is almost zero, see the short-dashed curve in Figure 8. Generalization increases the number of shared patterns, see the solid and long-dashed curves in Figure 8. The observed result can be explained by the fact that vocabularies of standalone model collections are specific and do not enable the inter-collection reuse of patterns. On the other hand, generalization, at the cost of information loss, facilitates knowledge reuse. For illustration purposes, we provide an excerpt of the shared action patterns for the case $minsup = 5$ and $minconf = 0.90$.

1. {move} ⇒ {act}
2. {act, move} ⇒ {test}
3. {improve, move} ⇒ {move}

The discovered examples illustrate the high degree of information loss. We further inspect the number of shared action patterns studying object-sensitive actions and, in particular, single-object action patterns. For this class we observe no shared patterns for original actions, and around 10 action patterns, once generalization is employed. Finally, for the class of behavioral action patterns no shared patterns are observed.

The study of action patterns shared by the SAP Reference Model and the Telecom collection reveals that their intersection is rather small. This stems from the fact that the collections are created independently: the vocabulary does not intersect. This means that synonyms might have been used for identical concepts in the two collections. This hypothesis is supported by the results obtained with action generalization. We were able to identify a set of action patterns, yet at the expense of information loss. On the other hand, our analysis might have also identified semantically wrong pattern matches across two collections. Yet, these are less likely to be found on a generalized level, but cannot be ruled out in an automatic fashion.

6. Related Work

The work presented in this article relates to different streams of research on business process modeling. We discuss patterns for business processes, intelligent modeling support, and research on activity labels.

There is a wide variety of *patterns* proposed for business processes and business process modeling. On the technical level, the workflow pattern initiative has identified various patterns for the specification of control flow [20], data flow [21], and resources [22] in workflow management systems. On a more conceptual level, Lonchamp proposed a set of collaboration patterns defining abstract building blocks for recurrent situations [23]. Tran et al. formalize process patterns using UML concepts [30]. Most closely related to our work is the research by Thom et al. [31]. The authors identify so-called *activity* patterns that specify eight different types of micro workflows, like approval or decision. Further, in [32] the authors describe a method for patterns derivation. While [32, 31] operates directly with activities, we use the concept of actions. As in real world models activities with different labels often signify the same action, usage of actions facilitates pattern derivation. Next, instead of direct analysis of a model graph structure, we rely on the concept of behavioral profiles. As relations captured by behavioral profiles are weaker than those defined by process models, [32] discovers only a subclass of behavioral action patterns.

The potential of improving business process modeling using *intelligent support and recommendations* has been recognized only recently. Hornung et al. define a concept to provide recommendations to the modeler based on search techniques [7]. The idea is to find similar models in the process repository and propose them as extensions to a process being currently modeled. This idea is in line with our approach, but requires a match not only in terms of actions, but also business objects and other textual content. We deem our approach to be more flexible and applicable across different modeling contexts. Further experiments are needed to check comparative strengths and weaknesses. A different stream of research investigates how far social software and Web 2.0 applications can provide recommendations to the modeler. Koschmider et al. propose a solution that enables collaborative modeling and user recommendations [33, 34]. In contrast to our work, the approach builds on behavior and suggestions of other modelers. Control flow correctness issues are addressed in [35], where the authors offer continuous verification of process models during modeling. In [36] the authors study how cooperative modeling is supported by fragment-driven modeling approach. However, this article primary focuses on describing the infrastructure for cooperative modeling, but not on the derivation of fragments (or action patterns). Gschwind et al. employ control flow patterns to accelerate business process modeling and minimize the number of modeling errors [37]. The authors develop a suggestion mechanism considering structural patterns and the model structure at hand. For a more general perspective, our work is related to reference modeling. Reference models are used as a means of assistance for adapting models in a particular domain [6]. Several conceptual approaches have been defined for customizing reference models to a specific application context, e.g. [38, 39, 40]. The action pattern approach aims to extract general process knowledge that can be used independent from a particular domain.

A recent stream of research works on the textual content of process models. These contributions are motivated by the fact that different grammatical styles of labeling activities in process models has a significant effect on usability [10]. Textual labels are also used for matching and comparing process models [7, 41]. Different approaches for matching corresponding activities of different models are integrated in the ICoP framework [42]. The content of activity labels is also used for defining suitable abstractions of a process model [43]. Natural language processing techniques are applied in [13]. The authors build up a descriptor space to assist in the creation of new process models. Such research builds on the efficient parsing of labels. Becker et al. reuse parsing techniques from computer linguistics for identifying the grammatical parts of

an activity label [44]. There is also work available to recognize the labeling style of an activity [5] and to automatically convert it from one style to another [12]. These approaches provide the preprocessing which is required for deriving action patterns automatically.

7. Conclusion

Large industrial process model collections put numerous model management challenges on their owners, yet create new opportunities. In this context, this article delivers a novel approach for externalizing hidden knowledge in such collections. This knowledge can be reused within various model management tasks. For instance, during the creation of a new process model, the designer may gain from suggestions on the missing activities. In this way it is possible to accelerate model design and assure model quality. The core concept of the proposed approach is an action pattern. Action patterns describe different relations among actions recurrently observed within a process model collection. The discovery of such relations is based on association rules mining. This article systematically explored the variety of dimensions along which action patterns are defined. We built upon our prior research [2, 3], which discussed two types of patterns, co-occurrence and behavioral, and different approaches to take objects into account, i.e., object-neutral and object-sensitive actions. In this article, we have integrated these dimensions into a framework that also considers the level of abstraction of actions and, therefore, facilitates pattern discovery. Navigation through our framework along these dimensions allows for the creation of action patterns of various types. Finally, we validated our approach applying it to industrial process model collections: the SAP Reference Model and the Telecom collection.

We foresee several directions for future work. First, this article assumes that a mapping of an activity label to an action is given. However, derivation of actions from labels is a challenging research topic and is in the focus of our work [5]. Bringing together label analysis and action pattern discovery would improve the applicability of action patterns. Further, this article discusses action generalization based on the MIT Process Handbook. However, we do not generalize objects. The study of object generalization by means of an ontology like WordNet [15], is the direct next step.

As process model collections are often incremented with new models, methods for action patterns derivation have to be efficient and adaptive. While in this article derivation of action patterns relies on Apriori algorithm, there is a potential to improve the performance by substituting it with a more efficient one. In addition, efficient strategies for adjusting the set of action patterns after creation of new process models have to be evaluated. Obviously, support and confidence values of existing patterns might easily be adapted. However, the detection of additional patterns, which have been ignored due to low support and confidence values before the model collection has been incremented, remains a serious issue. All these directions are rather unexplored for process models, and are on our future research agenda.

References

[1] M. Rosemann, Potential Pitfalls of Process Modeling: Part A, Business Process Management Journal 12 (2) (2006) 249–254.

[2] S. Smirnov, M. Weidlich, J. Mendling, M. Weske, Action Patterns in Business Process Models, in: ICSOC/ServiceWave, Vol. 5900 of LNCS, 2009, pp. 115–129.

[3] S. Smirnov, M. Weidlich, J. Mendling, M. Weske, Object-Sensitive Action Patterns in Process Model Repositories, in: Business Process Management Workshops, Vol. 66 of LNBIP, Springer, Hoboken, NJ, US, 2011, pp. 251–263.

[4] J. Mendling, Empirical Studies in Process Model Verification, Transactions on Petri Nets and Other Models of Concurrency 5460 (2009) 208–224.

[5] H. Leopold, S. Smirnov, J. Mendling, Recognising Activity Labeling Styles in Business Process Models, Enterprise Modelling and Information Systems Architectures 6 (1) (2011) 16–29.

[6] A. Scheer, Business Process Engineering: Reference Models for Industrial Enterprises, 2nd Edition, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1994.

[7] A. Koschmider, T. Hornung, A. Oberweis, Recommendation-based Editor for Business Process Modeling, Data & Knowledge Engineering 70 (6) (2011) 483–503.

[8] O. Holschke, Impact of Granularity on Adjustment Behavior in Adaptive Reuse of Business Process Models, in: BPM, Vol. 6336 of LNCS, Springer, 2010, pp. 112–127.

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

[10] J. Mendling, H. A. Reijers, J. Recker, Activity Labeling in Process Modeling: Empirical Insights and Recommendations, Information SystemsTo appear.
[11] P. Delfmann, S. Herwig, L. Lis, A. Stein, Eine Methode zur formalen Spezifikation und Umsetzung von Bezeichnungskonventionen für fachkonzeptionelle Informationsmodelle, in: MobIS, Saarbrücken, Germany, 2008, pp. 23–38.
[12] H. Leopold, S. Smirnov, J. Mendling, Refactoring of Process Model Activity Labels, in: NLDB, Vol. 6177 of LNCS, Springer, 2010, pp. 268–276.
[13] M. Lincoln, M. Golani, A. Gal, Machine-Assisted Design of Business Process Models Using Descriptor Space Analysis, in: BPM, Vol. 6336 of LNCS, Springer, 2010, pp. 128–144.
[14] T. W. Malone, K. Crowston, G. A. Herman, Organizing Business Knowledge: The MIT Process Handbook, 1st Edition, The MIT Press, Cambridge, MA, USA, 2003.
[15] A. G. Miller, Wordnet: A lexical database for English, Communications of the ACM 38 (11) (1995) 39–41.
[16] S. A. Caraballo, Automatic Construction of a Hypernym-labeled Noun Hierarchy from Text, in: ACL, 1999.
[17] R. Snow, D. Jurafsky, A. Y. Ng, Learning Syntactic Patterns for Automatic Hypernym Discovery, in: NIPS, 2004.
[18] C. Alexander, S. Ishikawa, M. Silverstein, A Pattern Language: Towns, Buildings, Construction, Oxford University Press, New York, 1977.
[19] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns, Addison-Wesley, Boston, MA, 1995.
[20] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, A. P. Barros, Workflow Patterns, Distributed and Parallel Databases 14 (1) (2003) 5–51.
[21] N. Russell, A. H. M. ter Hofstede, D. Edmond, W. M. P. van der Aalst, Workflow Data Patterns, Tech. Rep. FIT-TR-2004-01, Queensland University of Technology (2004).
[22] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, D. Edmond, Workflow Resource Patterns, Tech. Rep. WP 126, Eindhoven University of Technology (2004).
[23] J. Lonchamp, Process Model Patterns for Collaborative Work, in: Telecoop, 1998.
[24] J. Mendling, Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, Vol. 6 of LNBIP, Springer, 2008.
[25] R. Agrawal, T. Imielinski, A. N. Swami, Mining Association Rules between Sets of Items in Large Databases, in: COMAD, Washington, D.C., 1993, pp. 207–216.
[26] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules in Large Databases, in: VLDB, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994, pp. 487–499.
[27] M. Weidlich, J. Mendling, M. Weske, Efficient Consistency Measurement based on Behavioural Profiles of Process Models, IEEE Transactions on Software Engineering 37 (2011) 410–429.
[28] G. Decker, J. Mendling, Instantiation Semantics for Process Models, in: BPM, LNCS, Springer, 2008, pp. 164–179.
[29] J. Mendling, J. Recker, Towards Systematic Usage of Labels and Icons in Business Process Models, in: EMMSAD, Vol. 337, CEUR Workshop Proceedings, 2008, pp. 1–13.
[30] H. N. Tran, B. Coulette, B. T. Dong, Broadening the Use of Process Patterns for Modeling Processes, in: Software Engineering & Knowledge Engineering, Knowledge Systems Institute Graduate School, 2007, pp. 57–62.
[31] L. H. Thom, M. Reichert, C. M. Chiao, C. Iochpe, G. Hess, Inventing Less, Reusing More, and Adding Intelligence to BusinessProcess Modeling, in: DEXA, Vol. 5181 of LNCS, Springer, 2008, pp. 837–850.
[32] J. M. Lau, C. Iochpe, L. Thom, M. Reichert, Discovery and Analysis of Activity Pattern Cooccurrences in Business Process Models, in: ICEIS (3), Springer, 2009, pp. 83–88.
[33] A. Koschmider, M. Song, H. A. Reijers, Social Software for Modeling Business Processes, in: BPM Workshops, Vol. 17 of LNBIP, Springer, Milan, Italy, 2008, pp. 642–653.
[34] A. Koschmider, M. Song, H. A. Reijers, Advanced Social Features in a Recommendation System for ProcessModeling, in: BIS, Vol. 21 of LNBIP, Springer, Poznan, Poland, 2009, pp. 109–120.
[35] S. Kühne, H. Kern, V. Gruhn, R. Laue, Business Process Modelling with Continuous Validation, in: Business Process Management Workshops, Vol. 17 of LNBIP, 2009, pp. 212–223.
[36] K.-H. Kim, J.-K. Won, C.-M. Kim, A Fragment-Driven Process Modeling Methodology, in: ICCSA, Vol. 3482 of LNCS, Springer, 2005, pp. 817–826.
[37] T. Gschwind, J. Koehler, J. Wong, Applying Patterns during Business Process Modeling, in: BPM, Vol. 5240 of LNCS, Springer, 2008, pp. 4–19.
[38] M. L. Rosa, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, Questionnaire-based Variability Modeling for System Configuration, Software and System Modeling 8 (2) (2009) 251–274.
[39] H. A. Reijers, R. S. Mans, R. A. van der Toorn, Improved Model Management with Aggregated Business Process Models, Data & Knowledge Engineering 68 (2) (2009) 221–243.
[40] M. La Rosa, M. Dumas, A. ter Hofstede, J. Mendling, Configurable Multi-Perspective Business Process Models, Information Systems 36 (2).
[41] B. van Dongen, R. Dijkman, J. Mendling, Measuring Similarity between Business Process Models, in: CAiSE, Vol. 5074 of LNCS, Springer, 2008, pp. 450–464.
[42] M. Weidlich, R. M. Dijkman, J. Mendling, The ICoP Framework: Identification of Correspondences between Process Models, in: CAiSE, Vol. 6051 of LNCS, Springer, 2010, pp. 483–498.
[43] S. Smirnov, R. M. Dijkman, J. Mendling, M. Weske, Meronymy-Based Aggregation of Activities in Business Process Models, in: ER, Vol. 6412 of LNCS, Springer, 2010, pp. 1–14.
[44] J. Becker, P. Delfmann, S. Herwig, L. Lis, A. Stein, Towards Increased Comparability of Conceptual Models—Enforcing Naming Conventions through Domain Thesauri and Linguistic Grammars, in: ECIS, 2009, pp. 2636–2647.