Action Patterns in Business Process Models

Sergey Smirnov, Matthias Weidlich, Jan Mendling, Mathias Weske

Technische Berichte Nr. 30

des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Softwaresystemtechnik an der Universität Potsdam | 30

Sergey Smirnov | Matthias Weidlich | Jan Mendling | Mathias Weske

Action Patterns in Business Process Models

Universitätsverlag Potsdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de/ abrufbar.

Universitätsverlag Potsdam 2009
http://info.ub.uni-potsdam.de/verlag.htm

Am Neuen Palais 10, 14469 Potsdam
Tel.: +49 (0)331 977 4623 / Fax: 4625
E-Mail: verlag@uni-potsdam.de

Die Schriftenreihe **Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam** wird herausgegeben von den Professoren des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam.

Druck: docupoint GmbH Magdeburg

ISSN 1613-5652
ISBN 978-3-86956-009-0

Zugleich online veröffentlicht auf dem Publikationsserver der Universität Potsdam:
URL http://pub.ub.uni-potsdam.de/volltexte/2009/3358/
URN urn:nbn:de:kobv:517-opus-33586
[http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-33586]

Action Patterns in Business Process Models

Sergey Smirnov[1], Matthias Weidlich[1], Jan Mendling[2], and Mathias Weske[1]

[1] Hasso Plattner Institute, Potsdam, Germany
{sergey.smirnov,matthias.weidlich,mathias.weske}@hpi.uni-potsdam.de
[2] Humboldt-Universität zu Berlin, Germany
jan.mendling@wiwi.hu-berlin.de

Abstract. Business process management experiences a large uptake by the industry, and process models play an important role in the analysis and improvement of processes. While an increasing number of staff becomes involved in actual modeling practice, it is crucial to assure model quality and homogeneity along with providing suitable aids for creating models. In this paper we consider the problem of offering recommendations to the user during the act of modeling. Our key contribution is a concept for defining and identifying action patterns - chunks of actions often appearing together in business processes. In particular, we specify action patterns and demonstrate how they can be identified from existing process model repositories using association rule mining techniques. Action patterns can then be used to suggest additional actions for a process model. Our approach is challenged by applying it to the collection of process models from the SAP Reference Model.

1 Introduction

Business process management experiences a large uptake by the industry, as more and more companies analyze and improve their processes to stay competitive. Process models being formal representations of business processes facilitate many tasks in the domain of business process management. Thereby, instead of being an art of a few specialists, business process modeling becomes a daily routine of office staff. 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 [20]. For this reason, process modeling tools have to incorporate techniques to help these casual modelers to conduct their work in a productive way.

A research of business process modeling has revealed several approaches to make modeling more efficient. This research can be classified into two main categories. On the one hand, reference modeling aims to increase productivity based on the 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*. In contrast to well known workflow patterns, action patterns are closely related to semantic content of a process model. Meanwhile, unlike reference models, action patterns are abstract enough to be applicable in various domains. In this context, the term *action* essentially refers to the verb that describes the work content of a textual activity label.

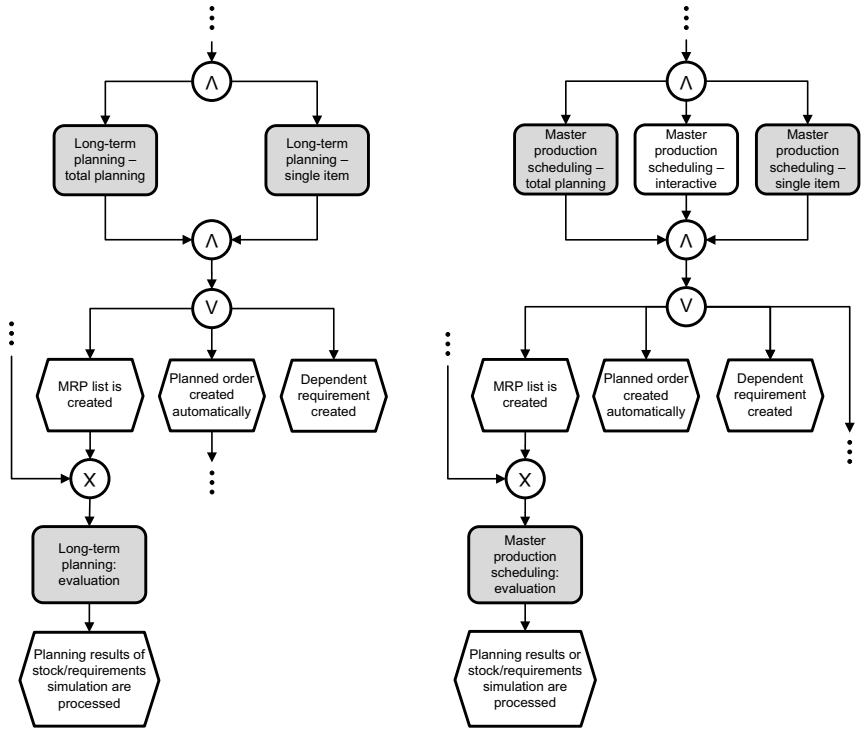
The contribution of this paper is a formal description of action patterns and an approach for identification of patterns in existing process model collections based on association rules mining. The mined action patterns can be used to suggest additional activities to the modeler during a modeling act. We specify two classes of patterns. Co-occurrence action patterns signify sets of actions that are likely to appear jointly in a model. Behavioral action patterns describe how co-occurring actions are related to each other in terms of behavioral constraints. This information allows us to identify the control flow position where an activity has to be added.

The rest of the paper is structured as follows. Section 2 provides a motivating example to illustrate our approach. Section 3 formalizes the action pattern concept and presents two classes of action patterns: co-occurrence action patterns and behavioral action patterns. Section 4 describes the evaluation of our approach by deriving action patterns from the SAP Reference Model. In Section 5 we present an outlook of the related work. Section 6 concludes the paper.

2 Motivating Example

An intrinsic complexity of business processes together with process models heterogeneity, originating from a variety of stakeholders and modeling purposes, calls for sophisticated support for process modeling. We distinguish two important drivers for such modeling support. On the one hand, the support aims at facilitating the design of a standalone process model. This kind of modeling support includes means to accelerate process model creation, assure correct model execution semantics, and increase model conciseness. However, the focus is purely on the *isolated* creation of a dedicated model: the application domain of this model is not taken into account. On the other hand, the rationale behind modeling support might be homogeneity of the modeling efforts. Process models created within a certain *domain*, might it be an organizational unit or a process model collection, should be modeled in a consistent and similar manner. In this case the emphasis is on avoiding redundancies and contradictions, as well as on enforcing modeling guidelines.

We illustrate the use case of *domain-aware* modeling support by means of the example in Fig. 1, which shows fragments of two EPCs from the SAP Reference Model [9]. Both business processes originate from the SAP material management and describe production planning. We see that the processes have a similar structure and semantics. For the long-term planning (Fig. 1(a)), as well as for



(a) Long-term planning (b) Master production scheduling

Fig. 1. Fragments of two similar planning processes from the SAP Reference Model

the master production scheduling (Fig. 1(b)), two similar planning steps are performed concurrently, and in both cases are succeeded by an evaluation. In Fig. 1 we highlight the activities which are interesting for us with grey color. Given these models, the creation of a model for a related process, e.g., a short-time production planning, might be supported as follows. After the modeler creates a function *Short-Term Planning - Total Planning*, we suggest to insert a concurrent function *Short-Term Planning - Single Item* on the fly. This recommendation can be derived from the analysis of the already existing models. We might also alert the modeler if he saves the model for short-term planning without having inserted a function for planning evaluation. The modeler might reconsider the modeling decisions and insert such a function, or rename an existing function, which has been intended to model the evaluation step, but was labeled differently (e.g., function *planning calculation* can be renamed to *planning evaluation*).

For obvious reasons, such *domain-aware* modeling support has to take into account semantics of existing process models. In this case semantics is not restricted to the model execution semantics. Instead, semantics has to be given in terms of concepts of the application domain. Applied to Fig. 1, a dependency between the planning steps (*total planning* vs. *single-item*), as well as their

relation to the planning evaluation, are examples of semantic dependencies specific for an application domain. The question how to derive a formalization of domain knowledge is crucial for a domain-aware modeling support.

To formalize domain knowledge, one might apply semantic annotations for all elements of the process. In this case, semantic information is represented in a structural way. This enables straight-forward processing and simplifies the usage for modeling support. However, this approach has an obvious drawback, since it requires semantic annotation of all model elements as a preliminary step. For a large collection of process models such a preliminary step requires enormous efforts and, thus, might not be feasible.

Thereafter, we follow another approach relying on the analysis of model element labels. The goal of this analysis is to understand the meaning of labels and extract domain knowledge out of an existing process model collection. Such an approach has to deal with the high ambiguity of a natural language. However, our experience (see [18]), as well as the experience of other researchers (see [4]), proves that label analysis is feasible. Besides, to simplify model comprehension for humans, modelers often stick to one schema when labeling model elements. An example is *verb + noun* schema employed for activity labeling. Analysis of such labels can be seen as an automated operation. The analysis outcome is the mapping of each activity to an *action* for which the activity stays in the model. For instance, from activity labeled with *send notification* one can derive that action *send* is performed over object *notification*.

In this paper we focus on supporting the modeler with recommendations on actions potentially missing in a process model. In general case, such recommendations depend on various model elements and other factors, but we focus on the analysis of activities. We are driven by two major reasons. First, giving recommendations on the missing actions (i.e., activities on the model level) requires exhaustive investigation of the existing ones. Second, a lion's share of business process model semantics is given by the activities. Hence, we formulate the recommendations based on the analysis of activities. To formalize the knowledge extracted from a process model collection we propose to use the notion of *action patterns*—groups of actions which often appear together in business processes. In the next section we elaborate on the concept of action patterns. However, before we proceed with action patterns discussion, we would like to summarize the assumptions used in this work:

Assumption 1 A process model collection is large enough to extract domain knowledge.
Assumption 2 An activity label signifies an action.
Assumption 3 There is a mechanism interpreting an activity label as an action.

3 Action Patterns

Pattern is a useful 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" [3]. While originally defined for architecture, this concept was adapted to software engineering in the 1990s (see [6]). In business process management, patterns have been defined, among others, for control flow [25], data flow [21], resources [22], and collaboration [14]. Also the MIT Process Handbook [15] can be related to the idea of describing a core solution to a recurring problem.

This section discusses the notion of actions patterns in order to meet the requirements for modeling support outlined above. First, Section 3.1 presents our formal framework for action patterns. Then, Section 3.2 defines co-occurrence action patterns. Finally, Section 3.3 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 alphabet 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 a finite set of nodes;
- $F \subseteq N \times N$ is the flow relation, such that (N, F) is a connected graph;
- $s \in A$ is the only start activity, such that $\bullet s = \emptyset$, where $\bullet n = \{n' \in N | (n', n) \in F\}$ for node n;
- $e \in A$ is the only end activity, such that $e\bullet = \emptyset$, where $n\bullet = \{n' \in N | (n, n') \in F\}$ for node n;
- $t : G \mapsto \{and, xor, or\}$ is a mapping that associates each gateway with a type;
- $l : A \mapsto \Gamma$ 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 of the model following on common execution semantics. Such semantics, in particular for the OR construct, has been presented in the existing work (see [16] as an example for EPCs).

To grasp the meaning of activities humans interpret their labels. In the context of this work interpretation of labels has great importance. Hence, we formalize it, introducing an alphabet of action terms $\mathcal{T}$ and a label interpretation function.

Definition 2 (Action Function). For a given process model $PM = (A, G, F, s, e, t, l)$, the *action function* $v : \Gamma \mapsto \mathcal{T}$ derives an action from a label. As a shorthand notation, we introduce $v_a : A \mapsto \mathcal{T}$ for deriving an action 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 actions of a process model.

Returning to the example with *send notification* label, the result of action function v is *send*. We also 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 is in the focus of data mining. 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 commerce 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 [1]. More advanced algorithms were presented in [2].

Further the generic formalism of association rule learning is adapted both for 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 [1], 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 $minconf$.

(a) Process model collection

Model	Actions
A	allocate analyze calculate collect evaluate settle summarize
B	allocate analyze asses calculate distribute entry evaluate reconcile repost settle split
C	allocate analyze calculate cost settle
D	allocate analyze calculate evaluate settle
E	allocate analyze collect calculate distribute evaluate settle summarize
F	allocate budget calculate copy define evaluate plan reconcile settle split transfer
G	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 $minsup = 5$

3.2 Co-occurrence Action Patterns

The first class of action patterns is co-occurrence action patterns. *Nomen est omen*, 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, while 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 paper we set our choice on Apriori algorithm, since it is efficient and simple [2]. In terms of large action sets this algorithm works as follows. As the 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. Table 1 illustrates the first (see Table 1(b)) and the second steps (see Table 1(c)) of Apriori work for the model collection captured in Table 1(a) given $minsup = 5$.

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 *minsup*.

3.3 Behavioral Action Patterns

Co-occurrence action patterns do not provide information about *how* the missing actions have to be introduced into the process model. As the next step, we consider action patterns that are enriched with information on relations between actions. First, we present preliminaries on behavioral relations and afterwards introduce the notion of a behavioral action pattern.

Behavioral Relations In order to capture behavioral aspects of a process on the level of pairs of activities, we apply the notion of *behavioral profiles* [28]. Such a profile consists of three relations that partition the cross product of all activities. Definition 1 does not specify execution semantics. However, for defining a behavioral profile, we impose syntactical requirements for the definition of all complete traces of a process model. That is, the (potentially unbounded) set of *complete process traces* $\mathcal{T}_{PM}$ for a process model $PM = (A, G, F, s, e, t, l)$ is a set of lists of the form $s \cdot A^* \cdot e$, such that a list entry contains the execution order of activities. Further on, we use $a \in \sigma$ with $\sigma \in \mathcal{T}_{PM}$ to denote that an activity $a \in A$ is a part of a complete 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* $y \not\succ_{PM} x$*;*
- *observation concurrency 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 Fig. 2. For instance, *(Template allocation)* $\rightsquigarrow$ *(Overhead calculation)* holds as there exists no trace, such that the latter function occurs before the former. With $\rightsquigarrow_{PM}^{-1}$ as the inverse relation for $\rightsquigarrow_{PM}$, *(Revaluation completed)* $\rightsquigarrow_{PM}^{-1}$ *(Template allocation)* also holds. It is worth to mention that $\rightsquigarrow_{PM}$,$\rightsquigarrow_{PM}^{-1}$,$+_{PM}$, and $||_{PM}$ partition the Cartesian product of activities $A \times A$ for a process model $PM = (A, G, F, s, e, t, l)$.

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 a designed 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 Fig. 2). 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.

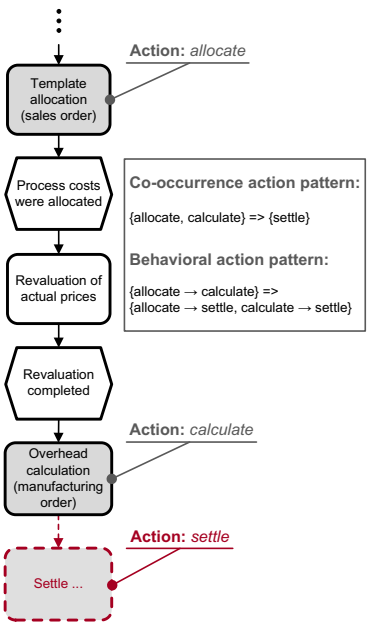


Fig. 2. Exemplary suggestion based on action patterns

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 more than in 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:
- 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. Provided $minsup$ and $minconf$ values, we can derive behavioral action patterns.

4 Evaluation based on the SAP Reference Model

To validate the proposed concepts and algorithms, we have conducted an experiment. The goals of the experiment were: 1) to check if it is possible to derive action patterns from a collection of process models and 2) to learn which support and confidence values are encountered in practice. The experiment consists of two parts: in the first part co-occurrence action patterns have been studied, in the second—behavioral action patterns.

The experiment studies the SAP Reference Model [9], a process model collection that has been used in several works on process model analysis [16]. 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, that are 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 first part of experiment for deriving co-occurrence patterns. In the second part, inspecting behavioral action patterns, the number of models was 421. The decrease in the model number is due to the exclusion of models with ambiguous instantiation semantics (cf., [5]) or behavioural anomalies (cf., [16]). At this stage we derived actions from activity labels manually. We foresee that this step can be automated in the future and are currently investigating techniques enabling the automation.

In the first part of the experiment we have derived 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 the pattern is more reliable, we aim to understand which values to be expected. In the SAP Reference Model the support value for all action sets is under 10, which is quite low given the fact that some actions appear several hundred times [17]. Next, it is hard to predict what is the minimally acceptable confidence level.

minconf \ *minsup*	2	3	4	5	6	7	8	9
0.50	522396	7395	2353	680	563	41	29	17
0.55	511373	6979	2247	665	550	34	23	13
0.60	510517	6123	2089	610	504	33	22	12
0.65	510498	6104	2070	591	497	26	16	9
0.70	484061	5569	1535	563	469	20	12	6
0.75	483415	4923	1477	505	421	19	11	6
0.80	483176	4684	1238	501	417	15	10	5
0.85	483135	4643	1197	460	417	15	10	5
0.90	483095	4603	1157	420	377	7	3	2
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

We conducted a set of experiments varying the level of support from 2 to 9 and 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 2, 17 patterns with support of 9, and not a single pattern has support 10. To illustrate how the derived patterns look like, we zoom into one cell of the table and list the patterns with $minsup = 7$ and $minconf = 0.95$:

- {pick} ⇒ {process}
- {level} ⇒ {evaluate}
- {permit} ⇒ {process}
- {archive enter} ⇒ {process}
- {allocate calculate} ⇒ {settle}

The results show that for the studied model collection the maximum support value is small. On the one hand, this is caused by unsystematic usage of labels: often the derived actions are semantically close, but are treated as different actions. An automatic clustering based on WordNet's synonym sets (see [19]) might help to achieve better support in the future. On the other hand, this fact can be explained by the heterogeneity of process models. The presence of process variants in the collection leads to the fact that some action patterns, especially of size 5-7, identify these variants in the model set.

Model \ Action pair	(allocate, calculate)	(allocate, settle)	(calculate, settle)
A	⇝	⇝	⇝
B	⇝	⇝	⇝
C	⇝	⇝	⇝
D	⇝	⇝	⇝

Table 3. Derived behavioral profiles for action set {allocate, calculate, settle}

(a) Action set {allocate, calculate, settle}

minconf \ *minsup*	2	3	4	5
0.50	12	12	12	0
0.55	12	12	12	0
0.60	12	12	12	0
0.65	12	12	12	0
0.70	12	12	12	0
0.75	12	12	12	0
0.80	12	12	12	0
0.85	12	12	12	0
0.90	12	12	12	0
0.95	12	12	12	0

(b) Action set {analyze, allocate, settle}

minconf \ *minsup*	2	3	4	5	6
0.50	170	12	4	2	0
0.55	161	12	4	2	0
0.60	157	8	3	2	0
0.65	157	8	3	2	0
0.70	130	8	3	2	0
0.75	129	7	2	1	0
0.80	129	7	2	1	0
0.85	129	7	2	1	0
0.90	129	7	2	1	0
0.95	129	7	2	1	0

Table 4. Dependency of behavioral action patterns number for 2 action sets on *minsup* and *minconf* values

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. In the experiment we considered those process models from the SAP Reference Model that can be mapped to free-choice Petri nets. Table 3 provides an example of relations for actions *allocate, calculate, settle*. Table 4(a) shows the number of patterns that can be derived for this set depending on the *minsup* and *minconf* values for relations. A more vivid example is the action set {*analyze, allocate, settle*}, for which the number of behavioral patterns varies greatly (see Table 4(b)). A concrete example of a behavioral action pattern which can be derived from Table 3 is {*allocate* $\rightsquigarrow$ *calculate*} $\Rightarrow$ {*allocate* $\rightsquigarrow$ *settle, calculate* $\rightsquigarrow$ *settle*}. This pattern prescribes that the three actions are sequentially constrained. They should appear in the process model such that first it is *allocated*, then *calculated*, and finally *settled*, which is a standard sequence of activities for financial assets.

5 Related Work

Our work can be related to different contributions to business process modeling. We focus on the three areas 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 [25], 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 [14]. Tran et al. formalize process patterns using UML concepts [24]. Most closely related to our work is the research by Thom et al. [23]. In their work, the authors identify so-called *activity* patterns

that specify eight different types of micro workflows like approval or decision. Our action pattern approach builds on the same observation that certain activities often occur jointly to achieve an over-arching goal. In contrast to this work, we do not assume a priori knowledge on which patterns might occur in a process, but focus on their discovery.

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 [8]. The idea is to find similar models in the process repository and propose them as extensions to the currently being modeled process. 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 in how far social software and interactive web applications (Web 2.0) can provide recommendations to the modeler. Koschmider et al. propose a solution that enables collaborative modeling and user recommendations [11, 12]. In contrast to our work, the approach builds on behavior and suggestions of other modelers. Control flow correctness issues are addressed in [13] where the authors offer continuous verification of process models during modeling. In [10] the authors study how cooperative modeling is supported by fragment-driven modeling approach. However, this paper 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 [7]. The authors develop a suggestion mechanism considering structural patterns and the model structure at hand. Also the work by Weber et al. on change patterns can be mentioned in relation to intelligent modeling support [27].

Recent contributions identify a textual analysis of activity labels as a important step to improve the pragmatic quality of process models. For instance, different labeling schemes and their impact on model understanding has been analyzed in [18]. Textual labels are also used for matching and comparing process models [8, 26]. Recent works by Becker et al. reuse parsing techniques from computer linguistics to efficiently identify the various parts of an activity label [4]. While we have derived the actions manually for our experiment reported in this paper, we are currently working on automating this step by using the approach taken by Becker et al.

6 Conclusion

In this paper we have addressed the challenge of assisting the designer in modeling a process. We defined the concept of action patterns, which capture co-occurrences of actions in existing process model collections. Our contribution is an approach based on association rules mining that identifies sets of actions that likely imply further actions. In this way, action patterns can be used to suggest additional

activities to the modeler. Furthermore, we utilize behavioral profiles to capture behavioral relations between co-occurring actions. Therefore, we also provide information on how an additional action should be included in the process model. Our approach has been validated using the SAP Reference Model.

At the present stage, we analyze activity labels to derive actions only. We do not consider business objects, which are also references in the labels of model elements. While this might be regarded as a limitation, we made this design choice to identify recurring patterns that hold for different business objects. Taking business objects into account offers several advantages, including object life cycle mining. A mined object life cycle is a helpful tool as it facilitates advanced modeling support. Another potential direction of the future work involves synonym recognition. Using thesauri like WordNet would allow to cluster actions that are closely related and gain stronger support for related patterns. In this context, it might also be possible to consider action hierarchies like the one developed for the MIT Process Handbook. Derivation of behavioral action patterns implies construction of behavioral profiles for the corresponding process models. To construct behavioral profiles with adequate performance, we restricted the set of investigated process models to those which can be mapped to free-choice Petri nets. The next step is to improve the underlying algorithms and widen the class of models which can be handled. All these directions are rather unexplored for process models, and are on our future research agenda.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In *COMAD*, pages 207–216, Washington, D.C., 1993.
2. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
3. Ch. Alexander, S. Ishikawa, and M. Silverstein. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, August 1977.
4. J. Becker, P. Delfmann, S. Herwig, L. Lis, and A. Stein. Towards Increased Comparability of Conceptual Models - Enforcing Naming Conventions through Domain Thesauri and Linguistic Grammars. In *ECIS*, June 2009.
5. Gero Decker and Jan Mendling. Instantiation semantics for process models. In Marlon Dumas, Manfred Reichert, and Ming-Chien Shan, editors, *BPM*, volume 5240 of *Lecture Notes in Computer Science*, pages 164–179. Springer, 2008.
6. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley, Boston, MA, January 1995.
7. Th. Gschwind, J. Koehler, and J. Wong. Applying Patterns during Business Process Modeling. In *BPM*, pages 4–19, Berlin, Heidelberg, 2008. Springer.
8. Th. Hornung, A. Koschmider, and G. Lausen. Recommendation Based Process Modeling Support: Method and User Experience. In *ER*, volume 5231 of *LNCS*, pages 265–278. Springer, October 2008.
9. G. Keller and T. Teufel. *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping*. Addison-Wesley, 1998.

10. K.-H. Kim, J.-K. Won, and C.-M. Kim. A Fragment-Driven Process Modeling Methodology. In *ICCSA*, volume 3482 of *LNCS*, pages 817–826. Springer, 2005.
11. A. Koschmider, M. Song, and H. A. Reijers. Social Software for Modeling Business Processes. In *BPM Workshops*, volume 17 of *LNBIP*, pages 642–653, Milan, Italy, September 2008. Springer.
12. A. Koschmider, M. Song, and H. A. Reijers. Advanced Social Features in a Recommendation System for Process Modeling. In *BIS*, volume 21 of *LNBIP*, pages 109–120, Poznan, Poland, April 2009. Springer.
13. S. Kühne, H. Kern, V. Gruhn, and R. Laue. Business Process Modelling with Continuous Validation. In *MDE4BPM*, pages 37–48, September 2008.
14. J. Lonchamp. Process Model Patterns for Collaborative Work. In *Telecoop*, 1998.
15. Th. W. Malone, K. Crowston, and G. A. Herman. *Organizing Business Knowledge: The MIT Process Handbook*. The MIT Press, Cambridge, MA, USA, 1st edition, September 2003.
16. J. Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, volume 6 of *LNBIP*. Springer, 2008.
17. J. Mendling and J. Recker. Towards Systematic Usage of Labels and Icons in Business Process Models. In *EMMSAD*, volume 337, pages 1–13. CEUR Workshop Proceedings, June 2008.
18. J. Mendling, H. A. Reijers, and J. Recker. Activity Labeling in Process Modeling: Empirical Insights and Recommendations. *Information Systems*, 2009. to appear.
19. A. G. Miller. Wordnet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
20. M. Rosemann. Potential Pitfalls of Process Modeling: Part A. *Business Process Management Journal*, 12(2):249–254, 2006.
21. N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Workflow Data Patterns. Technical Report FIT-TR-2004-01, Queensland University of Technology, 2004.
22. N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. Workflow Resource Patterns. Technical Report WP 126, Eindhoven University of Technology, 2004.
23. L. H. Thom, M. Reichert, C. M. Chiao, C. Iochpe, and G.N. Hess. Inventing Less, Reusing More, and Adding Intelligence to Business Process Modeling. In S. S. Bhowmick, J. Küng, and R. Wagner, editors, *DEXA*, volume 5181 of *LNCS*, pages 837–850. Springer, September 2008.
24. H. N. Tran, B. Coulette, and B. Th. Dong. Broadening the Use of Process Patterns for Modeling Processes. In *SEKE*, pages 57–62. Knowledge Systems Institute Graduate School, July 2007.
25. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003.
26. B. van Dongen, R. Dijkman, and J. Mendling. Measuring Similarity between Business Process Models. In *CAiSE*, pages 450–464, Berlin, Heidelberg, 2008. Springer.
27. B. Weber, M. Reichert, and S. Rinderle-Ma. Change Patterns and Change Support Features - Enhancing Flexibility in Process-aware Information Systems. *Data Knowl. Eng.*, 66(3):438–466, 2008.
28. M. Weidlich, J. Mendling, and M. Weske. Computation of Behavioural Profiles of Process Models. Technical report, Hasso-Plattner-Institute, June 2009. `http://bpt.hpi.uni-potsdam.de/pub/Public/MatthiasWeidlich/behavioural_profiles_report.pdf`.

Aktuelle Technische Berichte des Hasso-Plattner-Instituts

Band	ISBN	Titel	Autoren / Redaktion
29	978-3-940793-91-1	**Correct Dynamic Service-Oriented Architectures: Modeling and Compositional Verification with Dynamic Collaborations**	Basil Becker, Holger Giese, Stefan Neumann
28	978-3-940793-84-3	**Efficient Model Synchronization of Large-Scale Models**	Holger Giese, Stephan Hildebrandt
27	978-3-940793-81-2	**Proceedings of the 3rd Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering**	Hrsg. von den Professoren des HPI
26	978-3-940793-65-2	**The Triconnected Abstraction of Process Models**	Artem Polyvyanyy, Sergey Smirnov, Mathias Weske
25	978-3-940793-46-1	**Space and Time Scalability of Duplicate Detection in Graph Data**	Melanie Herschel, Felix Naumann
24	978-3-940793-45-4	**Erster Deutscher IPv6 Gipfel**	Christoph Meinel, Harald Sack, Justus Bross
23	978-3-940793-42-3	**Proceedings of the 2nd. Ph.D. retreat of the HPI Research School on Service-oriented Systems Engineering**	Hrsg. von den Professoren des HPI
22	978-3-940793-29-4	**Reducing the Complexity of Large EPCs**	Artem Polyvyanyy, Sergy Smirnov, Mathias Weske
21	978-3-940793-17-1	**"Proceedings of the 2nd International Workshop on e-learning and Virtual and Remote Laboratories"**	Bernhard Rabe, Andreas Rasche
20	978-3-940793-02-7	**STG Decomposition: Avoiding Irreducible CSC Conflicts by Internal Communication**	Dominic Wist, Ralf Wollowski
19	978-3-939469-95-7	**A quantitative evaluation of the enhanced Topic-based Vector Space Model**	Artem Polyvyanyy, Dominik Kuropka
18	978-3-939469-58-2	**Proceedings of the Fall 2006 Workshop of the HPI Research School on Service-Oriented Systems Engineering**	Benjamin Hagedorn, Michael Schöbel, Matthias Uflacker, Flavius Copaciu, Nikola Milanovic
17	3-939469-52-1 / 978-3-939469-52-0	**Visualizing Movement Dynamics in Virtual Urban Environments**	Marc Nienhaus, Bruce Gooch, Jürgen Döllner
16	3-939469-35-1 / 978-3-939469-35-3	**Fundamentals of Service-Oriented Engineering**	Andreas Polze, Stefan Hüttenrauch, Uwe Kylau, Martin Grund, Tobias Queck, Anna Ploskonos, Torben Schreiter, Martin Breest, Sören Haubrock, Paul Bouché
15	3-939469-34-3 / 978-3-939469-34-6	**Concepts and Technology of SAP Web Application Server and Service Oriented Architecture Products** (noch nicht erschienen)	Bernhard Gröne, Peter Tabeling, Konrad Hübner

ISBN 978-3-86956-009-0
ISSN 1613-5652