A Framework for Efficiently Mining the Organisational Perspective of Business Processes[☆]

Stefan Schönig[a], Cristina Cabanillas[b], Stefan Jablonski[a], Jan Mendling[b]

[a] *University of Bayreuth, Germany*
[b] *Vienna University of Economics and Business, Austria*

Abstract

Process mining aims at discovering processes by extracting knowledge from event logs. Such knowledge may refer to different business process perspectives. The organisational perspective deals, among other things, with the assignment of human resources to process activities. Information about the resources that are involved in process activities can be mined from event logs in order to discover resource assignment conditions, which is valuable for process analysis and redesign. Prior process mining approaches in this context present one of the following issues: (i) they are limited to discovering a restricted set of resource assignment conditions; (ii) they do not aim at providing efficient solutions; or (iii) the discovered process models are difficult to read due to the number of assignment conditions included. In this paper we address these problems and develop an efficient and effective process mining framework that provides extensive support for the discovery of patterns related to resource assignment. The framework is validated in terms of performance and applicability.

Keywords: Business process management, declarative process mining, event log analysis, organisational perspective, resource perspective

[☆] This work is funded by the "Europäischer Fonds für regionale Entwicklung" (EFRE) under grant 1502/89304-01/2012 (KpPQ) and the Austrian Research Promotion Agency (FFG) under grant 845638 (SHAPE).

[*] Stefan Schönig
Email address: stefan.schoenig@uni-bayreuth.de (Stefan Schönig)
URL: http://ai4.uni-bayreuth.de (Stefan Schönig)

1. Introduction

Business Process Management (BPM) is a well accepted method for structuring the activities carried out in an organisation, analysing them for efficiency and effectiveness, and identifying potential for improvement [1]. Processes are not always explicitly defined when the process models are designed. Actual process executions may constitute a valuable input for improving process design. Process mining provides methods for automatic process analysis, among others for discovering processes by extracting knowledge from event logs in form of a process model. Various algorithms are available to discover models capturing the control-flow of a process, related to the behavioural perspective of the process [2, 3]. For perspectives like the organisational perspective, which manages the involvement of human resources in processes, only partial solutions for mining have been developed despite the importance of resource information not only for performance but also for compliance analysis [4, 5, 6, 7].

The need to better support the organisational perspective was evidenced by previous approaches that mined this perspective [8, 9, 10, 11, 12, 13]. Prior work in this area focused on discovering specific aspects of the organisational perspective such as role models, separation of duty or social networks. However, comprehensive and integrated support for the well-established workflow resource patterns, and specifically in this context for the so-called creation patterns [14], was missing. Furthermore, the close interplay between the organisational and the behavioural perspectives was disregarded [15]. In [16] we addressed these gaps by developing a declarative process mining approach for the organisational perspective, which supports all the creation patterns as well as what we called cross-organisational patterns, which discover how the involvement of resources influences the control-flow of the process.

The research reported in this paper extends our prior work towards an efficient and effective mining framework. As illustrated in Figure 1, the framework is divided into an event log pre-processing phase, a phase for integrated resource mining including cross-perspective patterns, and a model post-processing phase.

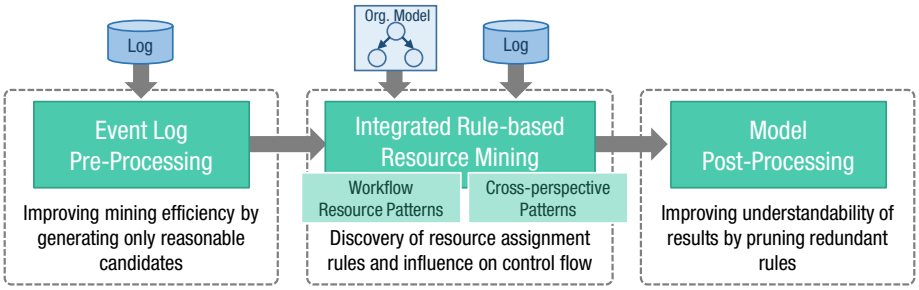


Figure 1: Framework for discovering resource-aware, declarative process models

We evaluate our approach with an implementation of the three phases; with simulation experiments for measuring performance; and with the application of the approach on a real-life event log for checking its effectiveness.

This research extends our previous work [16] as follows: *(i)* the developed pre-processing method increases the efficiency of the approach; *(ii)* the developed post-processing techniques increase the understandability of the results; *(iii)* a prototype of the entire framework has been implemented using Drools; and *(iv)* the approach has been extensively validated. In addition, the mining approach is explained in more detail. With our work, we complement research on process mining with an extensive support of the organisational perspective.

The remainder of this paper is structured as follows: Section 2 introduces background information. Section 3 describes our process mining approach. Sections 4 and 5 describe the event log preprocessing and postprocessing phases of the framework, respectively. Section 6 explains the evaluations performed. Section 7 describes the related work and Section 8 concludes the paper.

2. Background

In the following we introduce the concepts upon which our approach has been developed.

2.1. Organisational and Cross-Perspective Patterns in Processes

The well-known workflow resource patterns [14] capture the various ways in which resources are represented and utilised in business processes. Of specific interest to our research are the creation patterns since they describe different ways in which resources can be assigned to activities. These patterns, which will be referred to as *organisational patterns* from now on, include: *Direct Distribution*, or the ability to specify at design time the identity of the resource that will execute a task. *Role-Based Distribution*, or the ability to specify at design time that a task can only be executed by resources that have a given role. *Organisational Distribution*, or the ability to offer or allocate activity instances to resources based on their organisational position and their organisational relationship with other resources. *Separation of Duties*, or the ability to specify that two tasks must be allocated to different resources in a given process instance. *Case Handling*, or the ability to allocate all the activity instances within a given process instance to the same resource. *Retain Familiar* (a.k.a. *Binding of Duties*), or the ability to allocate an activity instance within a given process instance to the same resource that performed a preceding activity instance. *Capability-Based Distribution*, or the ability to offer or allocate instances of an activity to resources based on their specific capabilities. *Deferred Distribution*, or the ability to defer the specification of the identity of the resource that will execute a task until run time. *History-Based Distribution*, or the ability to offer or allocate activity instances to resources based on their execution history. Note that the creation patterns *Authorisation* and *Automatic Execution* are not in the list because they are not directly related to resource assignment.

It has been identified that process control-flow is intertwined with dependencies upon resource characteristics [15]. For instance, sometimes an activity must be executed eventually before another one for specific resources but not for others. As an example, resources with a certain role (e.g., trainees) must always perform a certain activity (e.g., double-check result) before they can continue with the following activity, but this might not be required for other roles (e.g., supervisors). We call this pattern *Role-Based Sequence*. A specific collection of

such *cross-perspective patterns* capturing these situations has not been defined. Nonetheless, in general, they can be defined by combining the aforementioned organisational patterns with the control-flow patterns described in [18]. The *Resource-Based Response* pattern, e.g., describes that for a special resource a certain activity has to follow eventually on another activity.

The organisational and the cross-perspective patterns constitute the set of patterns to be discovered by our framework.[1]

2.2. Event Logs for Mining the Organisational Perspective

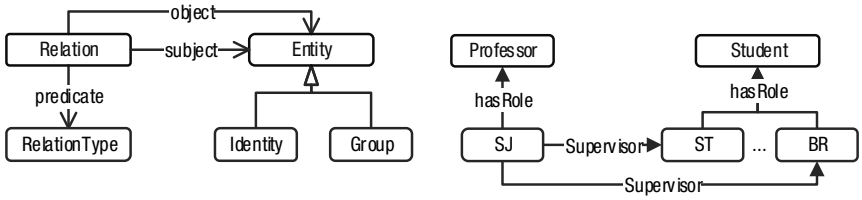
Our mining approach takes as input *(i) an event log*, i.e., a machine-recorded file that reports on the execution of tasks during the enactment of the instances of a given process; and *(ii) organisational background knowledge*, i.e., prior knowledge about the roles, capabilities and the membership of resources to organisational units, among others. In an event log, every process instance corresponds to a sequence (*trace*) of recorded entries, namely, *events*. We require that events contain an explicit reference to the enacted task and to the operating resource. Both conditions are commonly respected in real-world event logs [2]. For instance, the following excerpt of a business trip process event log encoded in the XES logging format [17] shows the recorded information of the *start event* of activity *Apply for trip* performed by resource *ST*.

```
<event>
   <string key="org:resource" value="ST"/>
   <date key="time:timestamp" value="2013-08-06T14:58:00.000+01:00"/>
   <string key="concept:name" value="Apply for trip"/>
   <string key="lifecycle:transition" value="start"/>
</event>
```

2.3. Representing the Output of the Mining

Since our aim is to discover the patterns explained in Section 2.1, the modelling language to represent the discovered processes must offer the possibility

[1] Therefore, when we speak about mining the organisational perspective we refer to both sets of patterns.



(a) Organisational meta model (b) Organisational model

Figure 2: Organisational meta model and example organisational model

to define *(i)* expressive organisational patterns and *(ii)* cross-perspective patterns. Two different representational paradigms for process models can be distinguished: procedural models describe which activities can be executed next in a process, and declarative models define by means of rules the execution constraints that the process has to satisfy [18]. Current procedural languages like Business Process Model and Notation (BPMN) [19] put a strong emphasis on control-flow and assume other perspectives to be specified separately. Cross-perspective patterns cannot be readily modelled. Declarative process modelling does not limit the number of perspectives involved in the constraints defined. However, a central shortcoming of existing languages like Declare [18] is that they are not provided with the capability to directly define the connection between the process behavior and other perspectives. We will use the Declarative Process Intermediate Language (DPIL) [20] for modelling the output of the mining because it supports multiple perspectives including the behavioural and the organisational perspectives, as well as the interplay between them. DPIL is expressive enough to cover the workflow patterns [20]. Nonetheless, the concepts of our approach are generic such that other declarative languages, such as Sciff [21] or LTL-based formalisms [22], could also be used as long as they provided support for the modelling of our target patterns.

In order to express organisational information, DPIL builds upon a generic organisational meta model [23] that is depicted in Figure 2a. It comprises the following elements: *Identity* represents agents that can be directly assigned

to activities, i.e., both human and non-human resources. *Group* represents abstract agents that may describe several identities as a whole, e.g., roles or groups. *Relation* represents the different relations (*RelationType*) that may exist between these elements. It is well suited for defining, e.g., that an identity has a specific role, that a person is the boss of another person, or that a person belongs to a certain department. In this context, relations are generally irreflexive. A relation is irreflexive if an identity cannot be in relation to itself. The *supervisor* relation, e.g., is irreflexive, since a person cannot be their own supervisor. In addition, some relations may be transitive. A relation is transitive if whenever an individual i_1 is related to another individual i_2 with that relation, and i_2 is in turn related to a third individual i_3 with the same relation, then i_1 is also related to i_3. For instance, the *supervisor* and *delegate* relations are typically transitive because organisations are usually hierarchically structured. Figure 2b illustrates an exemplary organisational model of a university research group, composed of two roles (Professor, Student) assigned to three people (SJ, ST, BR) and two relations between them indicating who is supervised by whom.

DPIL provides a textual notation based on the use of *macros* to define reusable rules. For instance, the *sequence*(a,b) macro states that the existence of a *start event* of task b implies the previous occurrence of a *complete event* of task a; and the *role*(a,r) macro states that an activity a is assigned to a role r. Figure 3 shows an example of a process for trip management modelled with DPIL. It specifies that it is mandatory to approve a business trip before flight tickets can be booked. Moreover, it is necessary that the approval be carried out by a resource with the role *Professor*.

3. Mining the Organisational Perspective

In this section we describe our approach to discover organisational and cross-perspective patterns. First, we describe how rule candidates are generated and checked. Then, we classify them according to support, confidence and interest factor values. Finally, we present a catalogue of rule templates that covers the

```
use group Professor
process BusinessTrip {
   task Book flight
   task Approve Application
   ensure role(Approve Application, Professor)
   ensure sequence(Approve Application, Book flight)
}
```

Figure 3: Process for trip management modelled with DPIL

target expressiveness (cf. Section 2.1).

3.1. Generation and Checking of Rule Candidates

Declarative process modelling languages like DPIL are based on so-called *rule templates*. A rule template captures frequently needed relations and defines a particular type of rules. Templates have formal semantics specified through logical formulae and are equipped either with user-friendly graphical representations (e.g., in Declare) or macros in textual languages (e.g., in DPIL). Unlike concrete rules, a rule template consists of placeholders, i.e., typed variables. A rule template is instantiated by providing concrete values for these placeholders. For instance, the model described in Section 2 makes use of two rule templates represented by the macros sequence(T_1,T_2) and role(T,G). These templates comprise placeholders of type *Task* T as well as *Group* G. In all well-known declarative process mining approaches, rule templates are used for querying the provided event log to find solutions for the placeholders. A solution is any combination of concrete values for the placeholders that yields a concrete rule that is satisfied in the event log. First, all possible rules need to be constructed by instantiating the given set of rule templates with all possible combinations of occurring process elements provided in the event log. For example, the sequence template consists of two placeholders of type $Task$. Assuming that $|T|$ different tasks occur in the event log, $|T|^2$ *rule candidates* are generated.

Let $|\Theta|$ be the number of different rule templates to be checked and $|P_j(i)|$ the number of different elements in the event log of a certain parameter type $P_j(i)$ contained in rule template θ_i. Let $k(i)$ be the number of placeholders in θ_i.

Trace	**start(of t_1)**	**direct(t_1,i_1)**
{**s**(t_2,i_1), **c**(t_2,i_2), **s**(t_3,i_1), **c**(t_3,i_1)}	false	false
{**s**(t_1,i_1), **c**(t_1,i_1), **s**(t_2,i_2), **c**(t_2,i_2), **s**(t_3,i_1), **c**(t_3,i_1)}	true	true
{**s**(t_1,i_1), **c**(t_1,i_1), **s**(t_3,i_3), **c**(t_3,i_3), **s**(t_2,i_2), **c**(t_2,i_2)}	true	true
{**s**(t_1,i_1), **c**(t_1,i_1), **s**(t_3,i_3), **c**(t_3,i_3), **s**(t_2,i_2), **c**(t_2,i_2)}	true	true
{**s**(t_1,i_4), **c**(t_1,i_4), **s**(t_3,i_1), **c**(t_3,i_1)}	true	false

Table 1: Event log and satisfaction of an example rule and its condition

The number of generated rule candidates $|R_{Cand}|$ is $|P_1(1)|\cdot|P_2(1)|\cdot...\cdot|P_{k(1)}(1)|$ + $|P_1(2)|\cdot|P_2(2)|\cdot...\cdot|P_{k(2)}(2)|$ + ... + $|P_1(i)|\cdot|P_2(i)|\cdot...\cdot|P_{k(i)}(i)|$ and therefore,

$$|R_{Cand}| = \sum_{i=1}^{|\Theta|}(\prod_{j=1}^{k(i)}|P_j(i)|) \tag{1}$$

The resulting candidates are subsequently checked w.r.t. the log. In many cases a rule candidate can be trivially valid. Consider the candidate *direct(t_1,i_1)*, i.e., *start(of t_1) implies start(of t_1 by i_1)*, which holds when task t_1 is performed by identity i_1, and the event log shown in Table 1. The notation used encodes the start and complete events of a specific task t performed by an identity i with s(t,i) and c(t,i), respectively. The given events are ordered temporally so that timestamps are not encoded explicitly. In the first trace the rule holds trivially because t_1 never happens. Using the terminology of [24], we say that the rule is *vacuously satisfied*. It is necessary to discriminate between traces in which a rule is trivially true and traces in which the rule is *non-vacuously satisfied*. Only the latter are considered interesting [25]. For first order logic rules that depict implications of the form $A \rightarrow B$, trivially and non-vacuously valid rules can be discriminated by additionally checking the condition A of the rule separately. Table 1 shows the results of checking the non-vacuous satisfaction of the *direct(t_1,i_1)* rule as well as its condition for each trace of the example log. In the first trace the rule is not (non-vacuously) satisfied because t_1 is never started, i.e., the condition is $false$. The rule holds non-vacously in the traces two to four. It is violated in trace five.

3.2. Metrics to Classify Rule Candidates

Checking rule candidates as described above provides for every candidate the number of instances, i.e., the traces in the event log where it non-vacously holds. Based on these values it is possible to classify rules and to separate non-valid from valid ones. Maggi et al. [24] adopted different metrics, specifically support (*supp*), confidence (*conf*) and interest factor (*int*) proposed by association rule mining for evaluating the relevance of rule candidates. Let $|\Phi|$ be number of traces in an event log Φ. Let $|\sigma_{nv}(r)|$ be the number of traces in which a rule $r : A \rightarrow B$ is non-vacously satisfied. The support $supp(r)$, confidence $conf(r)$ and $int(r)$ values of a rule r are defined as:

$$supp(r) := \frac{|\sigma_{nv}(r)|}{|\Phi|}, \;\; conf(r) := \frac{supp(r)}{supp(A)}, \;\; int(r) := \frac{supp(r)}{supp(A) \cdot supp(B)} \quad (2)$$

Considering again the event log of Table 1 and the direct(t_1,i_1) rule. Its support evaluates to $supp(r) = 0.6$, its confidence to $conf(r) = 0.75$ and its interest factor to $int(r) = 1.25$. We make use of the confidence value to classify a rule candidate r as a *valid* rule (i.e., satisfied in *almost all* traces) or a *non-valid* rule (i.e., violated in *most of* the recorded traces). Therefore, the threshold $minConf$ is introduced to classify rule candidates. Candidates r with $conf(r) > minConf$ are classified as valid. All rule candidates r with $conf(r) < minConf$ are non-valid rules and are not part of the resulting process model. Note that in case of rules that do not depict implications, the condition is satisfied in every trace; therefore, $supp(A) = 1$ and $conf(r) = supp(r)$. Using the confidence values of rule candidates it is directly possible to generate a DPIL process model reflecting organisational and cross-perspecitve patterns.

3.3. Rule Templates for Mining the Organisational Perspective

Since DPIL builds upon a flexible organisational meta model (cf. Section 2.3), it is possible to define rule templates that describe many aspects of the organisation. By instantiating these rule templates with all possible parameter combinations of defined resources, groups and relation types, it is possible to

generate rule candidates that focus on the organisational perspective of the process to be analysed. These candidates can then be checked under consideration of the event log and the organisational model.

In the following we define rule templates and their macros for our target set of patterns. First of all, we distinguish between templates for organisational patterns and templates for cross-perspective patterns. The former are, in turn, divided into two groups based on the types and number of parameters: rule templates related to a single task and rule templates related to more than one task. We provide representative examples for each group of rule templates that cover frequently needed organisational information. Note that besides the templates described next, further templates could be defined individually to cover the analyst's needs.

3.3.1. Rule Templates for the Assignment of Resources to a Single Task

This group includes rule templates that define organisational patterns referred to one process activity. The *Direct Distribution* pattern can be extracted with a *direct(T,I)* template. Given the free variables T and I and an event log with $|T|$ distinct tasks and $|I|$ distinct resources, there are $|T| \cdot |I|$ candidates to be checked.

```
direct(T,I) iff start(of T) implies start(of T by I)
```

The *Role-Based distribution* pattern can be extracted with a *role(T,G)* template. Here, rule candidates for every task and group combination are generated, i.e., $|T| \cdot |G|$ rule candidates need to be checked.

```
role(T,G) iff start(of T by :p) implies
              relation(subject p predicate hasRole object G)
```

The *Capability-Based distribution* pattern can be extracted with a *capability(T,RT,G)* template. A capability is represented by a relation of an individual to a group, e.g., i_1 *hasDegree ComputerScience*. According to the placeholders, $|T| \cdot |RT| \cdot |G|$ candidates are generated.

```
capability(T, RT, G) iff
start(of T by :p) implies relation(subject p predicate RT object G)
```

The assignment of resources based on organisational positions of individuals, described by the *Organisation-Based Distribution* pattern, can be extracted with an *orgDistSingle(T,RT,G)* template. Here, $|T| \cdot |RT| \cdot |G|$ rules must be checked.

```
orgDistSingle(T, RT, G) iff
start(of T by :p) implies relation(subject p predicate RT object G)
```

3.3.2. Rule Templates for the Assignment of Resources to Several Tasks

This group includes rule templates that define organisational patterns referred to several tasks. The *Separation of Duties* pattern can be extracted with a *separate(T_1,T_2)* template. For this template, $|T|^2$ candidates need to be checked.

```
separate(T1,T2) iff start(of T1 by :p) and start(of T2) implies
                start(of T2 by not p)
```

The *Retain Familiar* pattern can be extracted with a *binding(T_1,T_2)* template. Similarly to the previous case, $|T|^2$ candidates need to be checked.

```
binding(T1,T2) iff start(of T1 by :p) and start(of T2) implies
               start(of T2 by p)
```

The *Case Handling* pattern can be extracted with a *caseHandling* template. Here, $|T|$ candidates have to be checked.

```
caseHandling iff forall(task T start(of T) implies start(of T by :p))
```

Resources can also be assigned to tasks according to their organisational relation with the performers of other process activities, e.g., an approval task might be assigned to people that can supervise the work done by the performers of a previous task. This is covered by the *Organisation-Based Distribution* pattern and can be extracted with an *orgDistMulti(T_1,T_2,RT)* template where variable *RT* specifies the type of relation between the two individuals involved. There exist $|T|^2 \cdot |RT|$ rule candidates.

```
orgDistMulti(T1,T2,RT) iff start(of T1 by :p1) and start(of T2 by :p2)
                       implies relation(subject p1 predicate RT object p2)
```

3.3.3. Cross-Perspective Rule Templates

A cross-perspective rule describes a temporal dependency or constraint between tasks but only applies for a certain set of identities, like in the following examples. Note, that other well-known control-flow patterns described in [18] can be defined in a similar way. The *Role-Based Sequence* pattern can be extracted with a *roleSequence(T_1,T_2,G)* template. Here, $|T|^2 \cdot |G|$ candidates need to be checked.

```
roleSequence(T1,T2,G) iff start(of T2 by :p at :t) and
                      relation(subject p predicate hasRole object G)
                      implies complete(of T1 at < t)
```

The *Resource-Based Response* pattern can be extracted with a *resourceResponse(T_1,T_2,I)* template. In this case, $|T|^2 \cdot |I|$ candidates need to be checked.

```
resourceResponse(T1,T2,I) iff complete(of T1 by I at :t) implies
                                 start(of T1 at > t)
```

4. Pre-processing to Extract Meaningful Parameters

Real-life event logs and organisational models potentially contain a big set of distinct tasks, resources and groups. For instance, the BPI challenge 2011 event log of a hospital information system [26] contains 623 different tasks and 42 organisational groups. By only considering the *role* template, this already leads to $623 \cdot 42 = 26166$ candidates to be checked. Although many of these parameter combinations never occur together in the same trace, the corresponding rules need to be checked. This problem can also be observed when considering task-resource combinations of the event log in Table 1. Resource i_4 only occurs together with task t_1. Hence, candidates of the *direct* template where $I = i_4$ and $T \neq t_1$ are trivially true in all traces and can be neglected without checking.

The method proposed in [24] uses the well-known Apriori algorithm to pre-process the log and to extract *task combinations* that frequently occur together.

Rule Template	**Item**	**Itemset**
direct(T, I)	(Task, Identity)	L_1: {(Task, Identity)}
role(T, G)	(Task, Group)	L_1: {(Task, Group)}
capability(T, RT, G)	(Task, Group)	L_1: {(Task), (Group)}
orgDistS(T, RT, G)	(Task, Group)	L_1: {(Task), (Group)}
binding(T, T)	(Task)	L_2: {(Task), (Task)}
separate(T, T)	(Task)	L_2: {(Task), (Task)}
orgDistMulti(T, T, RT)	(Task, Task)	L_2: {(Task), (Task)}
roleSequence(T, T, G)	(Task, Group)	L_2: {(Task, Group), (Task, Group)}

Table 2: Required itemsets for exemplary organisational rule templates

The problem of mining frequent itemsets is to find all itemsets that satisfy a user-specified minimum *support*. The *support* of an itemset X is the percentage of traces that contain the items of X. Note that this support value is different from the one defined in Section 3.2, which depicts the fraction of traces where a certain rule is non-vacuously satisfied. Specifically, let $|\Phi|$ be the total number of traces recorded in the log. Let σ_X be the set of traces that contain a set of items X. The support value of an itemset X in Φ is defined as

$$supp(X) = \frac{|\sigma_X|}{|\Phi|}, where\ \sigma_X = \{\sigma \in \Phi | \forall_{x \in X}\ x \in \sigma\} \tag{3}$$

A task combination is considered to be relevant if it occurs in a sufficient number of traces, i.e., if its *support* value is greater than a given threshold *minSupp*. A *minSupp* of 0.05, e.g., claims that only rule candidates whose parameter combinations occur in at least 5% of the recorded traces are considered. We extended this method to also extract *task-resource* and *task-group* combinations that frequently occur together. In this way, it is possible to reduce the number of organisational rule candidates by ignoring infrequent parameter combinations. For instance, for the example log, only one out of three *direct(T,i_4)* candidates is generated and checked.

Table 2 shows the form of a single item and the required itemset for the already defined rule templates (cf. Section 3.3). Regarding the rule templates

for the assignment of resources to a single task, since only one task is involved, itemsets X with $|X| = 1$ are required. For instance, the *direct* template has two placeholders, one for tasks and one for identities and hence, itemsets of the form *(Task, Identity)* are needed. Regarding the rule templates for the assignment of resources to several tasks, since in all these templates two tasks are involved, itemsets with $|X| = 2$ are required. The *binding* template, e.g., takes frequent items of the form *(Task, Task)*. Finally, the cross-perspective rule templates also have two placeholders and hence, itemsets with $|X| = 2$ are required. The templates *capability*, *orgDistS* and *orgDistM* additionally contain a variable for a relation type. The amount of different relation types in organisational models, however, is usually insignificant compared to the number of different individuals and groups and can therefore be neglected.

5. Pruning of Discovered Models

The output of the mining phase is a process model with rules that state which resources are assigned to the process tasks, e.g., resources with specific roles or capabilities. The mining method extracts *all* the assignment rules related to each task. However, when several rules are extracted for one single task, not all of them might be strictly necessary to understand the process. Specifically, some rules may be implied by stronger rules because they are less restrictive and do not provide any value to the current resource assignment expression of a task. Those rules complicate the understandability of discovered models and hence, they are unnecessary. We identified two pruning approaches to eliminate unnecessary rules: pruning based on organisational rule hierarchies and pruning based on transitive reduction. The requirement for all pruning operations is that they do not change the meaning of the generated model.

5.1. Pruning based on Organisational Rule Hierarchies

Maggi et al. [27] proposed a technique to post-process a discovered model and to remove *weaker* rules if they are already implied by *stronger* rules only

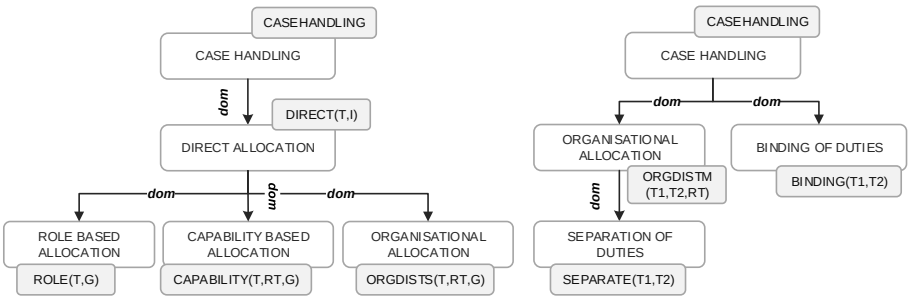
focusing on the hierarchy of control-flow templates. Hierarchies also exist in case of organisational rules.

We define rule hierarchies for the rule templates defined in Section 3.3. For that purpose, we introduce the *dominates* relation $\rightarrow_{dom}$ between two rules r_1 and r_2. Specifically, $r_1 \rightarrow_{dom} r_2$ means that rule r_1 is stronger than rule r_2. The defined rule hierarchies can then be used to prune and simplify discovered models. If a model contains two assignment rules r_1 and r_2 concerning the same task and $r_1 \rightarrow_{dom} r_2$, then r_2 can be pruned, i.e., removed from the model. User-defined rule types have to be integrated in exiting hierarchies by modelling experts. In order to justify the rule hierarchies described next, the following sets and functions must be introduced: $T = \{t_1, t_2, ..., t_n\}$ is a set of tasks; $R_i = \{r_1, r_2, ..., r_m\}$ is a set of assignment rules discovered for task t_i; $I = \{i_1, i_2, ..., i_p\}$ is a set of identities (i.e., individuals) of an organisation; $G = \{g_1, g_2, ..., g_q\}$ is a set of user groups of an organisation (e.g., roles); $id : R_i \rightarrow I$ returns the set of identities that meet the conditions defined by a rule; $pp : T \rightarrow I$ returns the set of potential performers of a task, where $pp(t_i) = \bigcap R_i$ and $pp(t_i) \neq \emptyset$ because otherwise rules would not have been extracted from the event log for task t_i; and $ap : T \rightarrow I$ returns the actual performer of a task for a specific task instance, so that $ap(t_i) \in pp(t_i)$.

We next explain how the rule hierarchies have been derived, providing a demonstration and an example for each *dominates* relation identified.

5.1.1. Rule Hierarchy for the Templates referred to a Single Task

We first focus on resource assignment rules for a single task (cf. Section 3.3.1), represented as $\Theta_1 = \{$*direct(T,I)*, *role(T,G)*, *capability(T,RT,G)*, *orgDistSingle(T,RT,G)*$\}$. Next, we describe and demonstrate the domination relations found out in Θ_1. For that, let us imagine that we have discovered two rules $R_1 = \{r_1, r_2\}$ for task t_1. Therefore, $pp(t_1) = id(r_1) \cap id(r_2)$, $pp(t_1) \neq \emptyset$. The aim in all cases is to prove that $id(r_1) \subseteq id(r_2)$, i.e., the individuals of r_1 are a subset of the individuals of r_2 and hence, r_2 is weaker and can be removed. The resulting rule hierarchy is visualised in Fig. 4a.



(a) Assignment rules w.r.t. a single task (b) Assignment rules w.r.t. two tasks

Figure 4: Hierarchies of organisational patterns

direct$(t_1,i_1) \rightarrow_{dom}$ ***role***(t_1,g_1), ***direct***$(t_1,i_1) \rightarrow_{dom}$ ***capability***(t_1,rt_1,g_1), ***direct***$(t_1,i_1) \rightarrow_{dom}$ ***orgDistS***(t_1,rt_1,g_1). *Direct* rules dominate *role* rules, *capability* rules and *orgDistS* rules. The demonstration of the three relations is the same, being $r_1 = direct(t_1,i_1)$ in all cases and $r_2 = role(t_1,g_1)$, $r_2 = capability(t_1,rt_1,g_1)$ and $r_2 = orgDistS(t_1,rt_1,g_1)$, respectively.

Proof. We demonstrate that $id(r_1) \subseteq id(r_2)$ by contradiction. Let $id(i_1) = \{r_1\}$ and $id(r_2) = \{i_2, i_3, i_4\}$, so $id(r_1) \not\subseteq id(r_2)$. That means i_1 does not have role g_1. Then, $pp(t_1) = id(r_1) \cap id(r_2) = \emptyset$, which is not possible by definition, as aforementioned. Therefore, and since $|id(r_1)| = 1$, $id(r_1) \subseteq id(r_2)$ is mandatory and hence, $pp(t_1) = id(r_1)$, which means r_2 is redundant and can be removed.

Example. Consider that a specific task *Book flight* has always been performed by a resource ST who has the role *Student* according to the organisational model. Then, the proposed method will (inevitably) discover rules *direct(Book flight,ST)* and *role(Book flight,Student)*. The identities derived from the latter rule are ST and BR. However, there is no evidence that BR can execute the task and hence, the *role* rule is not strong enough to be considered in the resource assignment.

role$(t_1,i_1) \not\leftrightarrow_{dom}$ ***capability***(t_1,rt_1,g_1), ***role***$(t_1,i_1) \not\leftrightarrow_{dom}$ ***orgDistS***(t_1,rt_1,g_1), ***capability***$(t_1,rt_1,g_1) \not\leftrightarrow_{dom}$ ***orgDistS***(t_1,rt_1,g_1). There is no domination relation between *role* and *capability* rules, *role* and *orgDist* rules, and *capability*

and *orgDist* rules. The demonstration is equivalent for any r_1 and r_2 belonging to these three groups.

Proof. The difference with respect to the previous demonstration lies on the cardinality of the rules involved. In this case, for any r_1, r_2 of one pair of rule types, $|id(r_1)| >= 1$ and $|id(r_2)| >= 1$. Since $id(r_1) \cap id(r_2) \neq \emptyset$, then either $id(r_1) \subseteq id(r_2)$ or $id(r_2) \subseteq id(r_1)$ depending on the number of individuals meeting the conditions specified by the rules. Therefore, a subsumption relation cannot be generalised and hence, both rules are, in general, necessary to calculate the potential performers of a task t_1, such that $pp(t_1) = id(r_1) \cap id(r_2)$.

Example. Consider the situation where the rules *role(Approve application,Professor)* and *capability(Approve application,hasDegree,CS)* have been extracted. It means that the task has been performed by someone with the role Professor and with a degree in Computer Science (CS). However, there might also be professors that do not have a degree in Computer Science, and vice versa. Therefore, to describe the necessary task condition, both rules are needed.

5.1.2. Rule Hierarchy for the Templates referred to Several Tasks

We now focus on resource assignment rules that involve two different tasks (cf. Section 3.3.2), represented as $\Theta_2 = \{$*binding(T_1,T_2)*, *separate(T_1,T_2)*, *orgDistMulti(T_1,T_2,RT)*$\}$. Next, we describe and demonstrate the domination relations found out in Θ_2. For that, let us imagine that we have discovered two rules $R_1 = \{r_1, r_2\}$ for task t_1, where one of the rules, in turn, refers to the assignment rule of task t_2. Similarly to the previous case, $pp(t_1) = id(r_1) \cap id(r_2)$, $pp(t_1) \neq \emptyset$. The aim is again to prove that $id(r_1) \subseteq id(r_2)$, i.e., the individuals of r_1 are a subset of the individuals of r_2 and hence, r_2 is weaker and can be removed. The resulting rule hierarchy is visualised in Fig. 4b.

separate(t_1,t_2) $\not\leftrightarrow_{dom}$ ***binding(t_1,t_2)***. There is no domination relation between *separate* and *binding* rules.

Proof. The demonstration is a contradiction by definition. The *separate* rule implies that $\forall ap(t_1), \forall ap(t_2)$ in a specific process instance, $ap(t_1) \neq ap(t_2)$,

i.e., both tasks have always been performed by different identities. The *binding* rule, however, states that $\forall ap(t_1), \forall ap(t_2)$ in a specific process instance, $ap(t_1) = ap(t_2)$, i.e., both tasks have always been performed by the same identity. In case both rules were extracted for task t_1, $id(r_1) \cap id(r_2) = \emptyset$ and hence, $pp(t_1) = \emptyset$. Therefore, these two rules can simply never be extracted at the same time because they are mutually exclusive.

orgDistMulti$(t_1,t_2,rt_1) \not\leftrightarrow_{dom}$ ***binding***(t_1,t_2). There is no domination relation between $orgDistMulti$[2] and *binding* rules.

Proof. Similarly to the previous case, the demonstration is a contradiction by definition. With an $orgDistMulti$ rule using an irreflexible relation, $ap(t_1) \neq ap(t_2)$. However, according to the *binding* rule, $ap(t_1) = ap(t_2)$. Hence, rules of these two types will never be extracted at the same time because they are mutually exclusive.

Example. Consider the situation where the rules *orgDistMulti(Approve application,Apply for trip,supervisor)* and *binding(Approve application,Apply for trip)* have been extracted for a task. It means that the application must be approved by the supervisor of the person who applies for the trip. Since a person cannot be a supervisor of herself, the tasks are performed by different individuals. However, according to the second rule, the two tasks should be performed by the same person.

orgDistMulti$(t_1,t_2,rt_1) \rightarrow_{dom}$ ***separate***(t_1,t_2). $orgDistMulti$ rules dominate *separate* rules.

Proof. Let $r_1 = orgDistMulti(t_1,t_2,rt_1)$ and $r_2 = separate(t_1,t_2)$. Assuming irreflexible relations in the organisation, according to both rules $ap(t_1) \neq ap(t_2)$. Since $id(r_1) \subseteq id(r_2)$, $pp(t_1) = id(r_1)$, which means r_2 is redundant and can be removed.

Example. Consider the situation where the rules *orgDistMulti(Approve application,Apply for trip,supervisor)* and *separate(Approve application,Apply for*

[2]Note that we assume that all relations are irreflexive (cf. Section 2).

trip) have been extracted for a task. It means that the application must be approved by the supervisor of the person who applies for the trip. Since a person cannot be a supervisor of herself, the tasks are performed by different individuals. However, not all the other persons in the organisation might be supervisors of the person applying for the trip. Therefore, this condition is more restrictive than the separation of duties and then, the latter is not necessary in the resource assignment expression.

5.1.3. Rule Hierarchy for the Cross-Perspective Templates

Finally, we address cross-perspective rules (cf. Section 3.3.3), represented as $\Theta_3 = \{roleSequence(T_1,T_2,G),\ resourceSequence(T_1,T_2,I)\}$. Notice that in this case the approach is different from Θ_1 and Θ_2 since we aim at generalising under which conditions a specific activity order must take place. That means that a rule r_1 is stronger than a rule r_2 if $id(r_2) \subseteq id(r_1)$. As demonstrated next, $roleSequence(t_1,t_2,g_1) \rightarrow_{dom} resourceSequence(t_1,t_2,i_1)$.

Proof. Let us imagine that we have discovered two rules $R_1 = \{r_1, r_2\}$, where $r_1 = roleSequence(t_1,t_2,g_1)$[3] and $r_2 = resourceSequence(t_1,t_2,i_1)$. The temporal dependency is the same in both cases, specifically, a specific task order determined by $sequence(t_1,t_2)$. Therefore, we could assume that $r_1 = role(t_1,g_1)$ and $r_2 = direct(t_1,i_1)$. According to the aforementioned criterion, since $|id(r_1)| >= 1$ and $|id(r_2)| = 1$, $id(r_2) \subseteq id(r_1)$, i.e., the individuals of r_2 are a subset of the individuals of r_1 and hence, r_2 is weaker and can be removed.

Example. Consider that task *Apply for trip* has always been performed before task *Book flight* when executed either by resource ST or by resource BR, who have the role *Student* according to the organisational model. Then, the proposed method will (inevitably) discover rules *resourceSequence(Apply for trip,Book flight,ST)*, *resourceSequence(Apply for trip,Book flight,BR)* and *roleSequence(Apply for trip,Book flight,Student)*. Since the individuals of both

[3]Note that to discover a *roleSequence* it is necessary to identify at least two entries in the log in which different resources with the same role are associated to a specific task sequence.

resourceSequence rules (i.e., ST and BR) are a subset of the individuals of the *roleSequence* rule, they can both be removed from the model.

5.2. Pruning based on Transitive Reduction

The assignment rules in Θ_2 (cf. Section 5.1.2) may be affected by transitivity. In particular, redundancy may be caused by the interplay of three or more rules of the same type applied to different activities. Consider a set of discovered *binding* rules, such as *binding*(t_1,t_2), *binding*(t_2,t_3) and *binding*(t_1,t_3). Here, the rule between t_1 and t_3 is redundant because it belongs to the transitive closure of the other rules. In other words, if task t_1 has always been performed by the same resource as t_2, and task t_3 has always been performed by the same resource as t_2, then also t_1 and t_3 have been performed by the same resource. Therefore, *binding*(t_1,t_3) is unnecessary and could be removed using the transitive reduction algorithm as defined in [28]. *OrgDistMulti* rules can be transitively reduced in a similar way if they refer to the same relation type rt and if rt is a transitive relation (cf. Section 2). However, *separate* rules are not transitive, i.e., if t_1 is not performed by the same resource as t_2 and t_2 is not executed by the same resource as t_3, then we cannot conclude that t_1 is also not performed by the same resource as t_3.

6. Evaluation

We evaluate our framework in three steps. We first describe how it has been implemented. We then show its efficiency with simulation experiments. Finally, we report on the results of applying the framework on a real-life event log.

6.1. Implementation

The problem of checking a large set of rule candidates can be solved by efficient pattern matching methods like the *rete algorithm* [29]. Instead of checking each rule separately, the rete algorithm first identifies common parts of the provided set of rules and constructs a *rete network*. Based on this decision network, common rule parts just need to be checked once. The JBoss

Drools platform[4] provides a current implementation of this method. In order to check rule candidates with Drools, they are translated into the Drools Rule Language (DRL). Like in DPIL, rules in DRL consist of a condition (*when* part) and a consequence (*then* part). If the condition holds, the consequence will be performed. DRL supports language elements to describe rules of first order logic, hence being equivalent to DPIL. The transformation of the most important expressions from DPIL to DRL are shown in Table 3. DPIL rules are translated into DRL rules like in row 3. As can be seen, the complete DPIL rule is placed in the ***when*** part of the DRL rule. The consequence, i.e., the ***then*** part, only contains a procedure call that signals the satisfaction of the corresponding rule to the program environment (*listener*). Since DRL does not support a logical implication directly, DPIL implications must be translated into DRL according to the logical equivalence $A \rightarrow B \equiv \neg(A \wedge \neg B)$ (cf. row 4 in Table 3). The described approach has been implemented in the *DpilMiner* application[5].

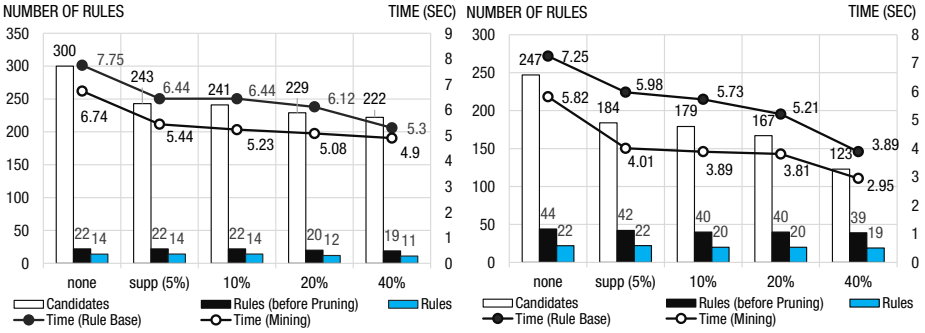
6.2. Performance Evaluation

To analyse performance we used the DpilMiner with different configurations using an event log of a university business trip management system[6]. The log contains 2104 events of 10 different activities related to the application

[4]Documentations about JBoss Drools is available at `http://docs.jboss.org/drools`
[5]A screencast of the DpilMiner is accessible at `http://www.kppq.de/miner.html`
[6]The event log is available for download at `http://workbench.kppq.de`

Nr.	DPIL expression	DRL expression
1	task T :t	$t: Task(id == "T")
2	start(**of** T)	$t: Task(id == "T") **and** Start(Task == $t)
3	expr	**rule** Id
		when expr **then** listener.onRuleOccured(drools.getRule()));
4	x **implies** y	**not** (x **and not** y)

Table 3: Rules for transforming DPIL to DRL expressions



(a) Results using rule template set 1 (b) Results using rule template set 2

Figure 5: Performance evaluation using different sets of rule templates

and the approval of university business trips as well as the management of accommodations and transfers, e.g., booking hotels and transport tickets. The system has been used for 6 months by 11 employees of a research institute of the University of Bayreuth (Germany). The organisational model of the institute assigns the 11 identities to 4 distinct roles, specifically 6 PhD students, 1 professor, 1 secretary and 3 administration employees. In total, there are 128 business trips, i.e., traces, recorded. All the computation times reported in this section are measured on a Core i7 CPU @2.80 GHz with 8 GB Ram.

Our approach has been tested with two different sets of rule templates. Fig. 5a shows the results of applying the approach with template *set* 1, which contains the templates *direct*, *role*, *binding* and *orgDistMulti*. Fig. 5b shows the results for template *set* 2, which contains the *sequence* template and the *roleSequence* cross-perspective template.

We analysed the time to build the rete network, i.e., the rule base[7], as well as the time to perform the actual mining process taking into account a different number of rule candidates. This was achieved by considering different *minSupp* values during the pre-processing phase ranging from 0 to 0.4 (cf.

[7]Note that the rule base only needs to be built once for different applications since the set of candidates depends on the occurring entities and not on the number of events or traces.

Section 4). The analysis shows the feasibility of our approach since in both tests, despite a big amount of candidates, only a manageable number of rules has been discovered. Especially the diagram in Fig. 5b highlights the benefit of the pre-processing approach. With increasing $minSupp$, the number of candidates to check considerably decreases, which reduces the processing time up to 50%. However, almost the same number of rules has been discovered in all cases before the post-processing phase. However, both diagrams show that the number of extracted rules is clearly reduced by pruning unnecessary rules. Fig. 5b, e.g., shows that the number of rules can be reduced by 50%.

In order to check the efficiency of our approach we also applied the implementation of the DeclareMiner [30] available in the Process Mining Framework (ProM) by only analysing the *precedence* template of Declare [18], which equates to the *sequence* template of DPIL. With standard settings, the DeclareMiner needed 14.85 sec to analyse the provided event log with the *precedence* template. Even if we analysed the example log with 2, respectively 4, rule templates, our approach was still faster in any case. For template *set* 1 and without pre-processing, the generation of the rule base for the rete algorithm took 7.75 sec while the actual analysis took only 6.74 sec.

6.3. Application to Real-Life Event Log

In this section we describe our findings when applying the approach to the university business trip log of Section 6.2. We analysed the log with the 6 aforementioned rule templates. With $minSupp = 0.1$ in the pre-processing phase and after removing unnecessary rules in the post-processing phase, we extracted 34 rules in total. The extracted resource assignment rules are composed of 4 *direct*, 1 *role*, 5 *binding* and 4 *orgDistMulti* rules. The rules with control-flow information include 14 *sequence* and 6 *roleSequence* rules. For the classification in satisfied and violated rules, we used $minConf = 0.85$ and $minInt = 1.0$. For space reasons, we only describe some interesting parts of the resulting model (cf. Figure 6). The discovered model shows that task "Approve Application" has mostly been performed by the identity "SJ" (*direct*). Furthermore, "Check

```
ensure direct(Approve Application, SJ)
ensure role(Check Application, Administration)
ensure binding(Apply for trip, Book flight)
ensure binding(Apply for trip, Book accommodation)
ensure binding(Apply for trip, Book transfer)
ensure orgDistMulti(Approve Application, Apply for trip, supervisor)
ensure roleSequence(Apply for trip, Book flight, Student)
```

Figure 6: Examples of discovered rules

Application" has mostly been performed by a resource with the role "Administration" (*role*). The three binding of duties rules show that the resource who booked the flight tickets, the accommodation and the transfer service has to be the person that applies for the trip (*binding*). Moreover, the resource who approves the trip application is the supervisor of the applicant (*orgDistMulti*). Regarding cross-perspective patterns, there are cases in which certain employees already booked a flight without applying for the trip. However, when analysing the task order under consideration of performing resources, we extracted that students always applied for the trip before they booked the flight (*roleSequence*).

In a second step we evaluated the quality of the mining results and how varying the mining configuration, i.e., different thresholds, influences it. Therefore, three discovered models (M1, M2, M3) based on different configurations of the approach on the same event log were discussed and evaluated in a workshop. The models were extracted using different $minSupp$ values during the pre-processing phase as well as different $minConf$ values during the mining phase. Table 4 shows the characteristics of the discovered models. M1 has been discovered by applying the approach without any pre-processing (*low filtering*). M2 depicts the model that has been described before and is based on a pre-processed log with *minSupp=0.1* (*medium filtering*). Both M1 and M2 include rules r with $conf(r) > 0.85$. One task that occurs in less than 10% of traces and the corresponding rules have been filtered in M2. M3 is based on $minConf = 0.9$, i.e., less rules are classified as satisfied (*high filtering*). The workshop was carried out with 8 process participants, i.e., university employees that represented all

the organisational groups involved. After we provided a general overview about the process and the workshop setting, each of the extracted rules was classified by the participants.

For evaluating the quality of the results, we rely on standard metrics from information retrieval precision and recall [31]. The harmonic mean (F-measure) of precision and recall is an adequate value for measuring the overall quality of extracted models [31]. To compute recall and precision, rules have been classified into one of three categories, i.e., *(i)* true-positive (T_P : correctly discovered); *(ii)* false-positive (F_P : incorrectly discovered); *(iii)* false-negative (F_N: incorrectly missing). Precision, recall and F-measure are defined as follows:

$$Precision = \frac{T_P}{T_P + F_P}, \qquad Recall = \frac{T_P}{T_P + F_N}, \qquad F = 2 \cdot \frac{P \cdot R}{P + R} \tag{4}$$

The results of the workshop as well as the calculated quality metrics are collected in Table 4. First of all, we focus on the results of M2. According to

	Model 1	**Model 2**	**Model 3**
Mining configuration	Low filtering	Medium filtering	High filtering
minSupp **(Pre-processing)**	⊘	0.1	0.1
minSupp	⊘	0.2	0.2
minConf	0.85	0.85	0.9
Characteristics of models			
Number of tasks	10	9	9
Number of identities	10	10	10
Number of rules	47	39	31
Metrics			
T_P (*correctly discovered*)	40	34	28
F_P (*incorrectly discovered*)	7	5	3
F_N (*incorrectly missing*)	0	6	12
Precision	0.85	0.87	0.9
Recall	1.0	0.85	0.7
F-measure	**0.92**	**0.86**	**0.8**

Table 4: Characteristics, results and metrics of discovered models

the information gathered from the process participants, 34 of 39 rules have been classified as relevant (T_P) while 5 rules have been discovered incorrectly (F_P). Furthermore, 6 missing rules (F_N) have been identified in the discussion. The reason is that a task and the assignment rules related to that task were filtered in the pre-processing phase. Based on this classification, we obtain *Precision=0.87*, *Recall=0.85* and therefore, *F=0.86*. Comparing the three models in Table 4 we can observe that M1 has the highest F-measure, i.e., the best quality. Since the model was extracted without filtering infrequent behaviour, there were no rules missing (*Recall=1.0*). Without filtering, however, M1 also contains 7 irrelevant rules leading to a lower precision value. Since M3 is based on a higher $minConf$ threshold, the model contains fewer rules. However, some of the missing rules were identified as relevant by the workshop participants. Due to the missing rules, M3 features the lowest recall and thus also the lowest F-measure.

7. Related Work

Several approaches have been proposed in the literature for the discovery of declarative process models. In [25] the authors present an approach that allows the user to select from a set of predefined Declare templates the ones to be used for the discovery. Maggi et al. propose an evolution of this approach in [24] to improve performance by pre-processing the event log with frequent pattern mining techniques. Other approaches to improve the performance of process mining are presented in [3, 32]. Additionally, there are post-processing approaches that aim at simplifying the resulting Declare models in terms of redundancy elimination [33] and disambiguation [27]. The approach proposed in [34] allows for the specification of rules that go beyond the traditional Declare templates. In [35], an approach for analysing event logs with Timed Declare, an extension of Declare that relies on timed automata, is described. The work in [36] first covered the data perspective in declarative process mining, although this approach only allows for the discovery of discriminative activation conditions. In essence, the focus of the aforementioned approaches is control-flow with extensions to

cover data without analysing resource-related information.

Complementary to them are techniques for mining the organisational perspective of a process [33]. Methods for analysing event logs w.r.t. resources are mainly focused on enriching a given *procedural* model with resource assignments [13]. Several methods focus on extracting an organisational model [9] or a social network [8]. There are also approaches that analyse the influence of resources on process performance [10]. However, the approaches that are of highest interest to us are those collected in Table 5, which address the discovery of organisational or cross-perspective patterns. Staff assignment mining [37] is able to extract complex assignment rules based on decision tree learning. However, the resource assignments are only related to one single task (cf. Section 5.1.1). Works on role mining [11, 12] are, on the contrary, interested in those types of rules referring to several tasks (cf. Section 5.1.2) but disregard other patterns. Resource mining is also implemented in ProM. In [38] the authors propose a two-step technique for enriching a given control-flow model with swimlanes based on the *Handover of Roles (HooR)* principle[8]. In the first step the pairs of immediately consecutive activities are analyzed in terms of potential role changes based on three rules: *(i)* pairs of immediately consecutive activities that are always executed by the same resource do not involve a HooR, *(ii)* pairs of immediately consecutive activities that are each executed by exactly the same set of resources do not involve a HooR and *(iii)* pairs of immediately consecutive activities that are, to a certain proportion w, executed by the same resources do not involve a HooR. All rules are based upon the assumption that each resource has exactly one role. The clustering-inspired algorithm generates a partition of activities for each HooR. The last step of the algorithm merges similar partitions in order to identify the actual roles. Finally, the algorithm chooses the most suitable final partitioning based on an entropy measure.

None of the aforementioned approaches on resource mining covers the whole sets of organisational and cross-perspective patterns that constitute the goal

[8]For space limitations, we refer to [38] for details on this principle.

of our work. The DpilMiner was developed to bridge that gap and hence, we used its mining approach [16] for the mining phase of our framework, which we extended with pre-processing and post-processing techniques inspired by the solutions related to mining the process control-flow.

8. Conclusions and Future Work

In this paper we presented a process mining framework to discover resource-aware process models. Our approach is based upon the mining approach introduced in [16], which we extended with pre-processing and post-processing phases. This increased efficiency while generating simplified process models that provide the same valuable information, as demonstrated by our evaluations.

Since our approach relies on DPIL [20], the mining capabilities are limited to its expressiveness. Therefore, inter-case dependencies, such as those represented in the *History-Based Distribution* pattern, cannot be discovered. It is an interesting question for future research how such dependencies can be mined and effectively depicted in a process model. Furthermore, there might be more ways to prune discovered models that take into account more knowledge besides hierarchies and transitive reduction. By pruning more intelligently, a better model

Pattern	**Mining approach**
Direct Distribution	[9, 12, 16, 37, 39, 38]
Role-based Distribution	[9, 12, 16, 37, 39, 38]
Deferred Distribution	-
Separation of Duties	[11, 12, 16]
Case Handling	[9, 12, 16]
Retain Familiar	[11, 12, 16]
Capability-based Distribution	[37, 16]
History-based Distribution	-
Organisational Distribution	[37] (single task) [16] (incl. several tasks)
Cross-Perspective Patterns	[16]

Table 5: Existing approaches for mining the organisational perspective

could be obtained. Finally, we plan to investigate options for mapping the output to graphical process modelling notations to increase readability.

References

[1] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer-Verlag Berlin Heidelberg, 2013. doi:10.1007/978-3-642-33143-5.

[2] W. van der Aalst, Process mining: discovery, conformance and enhancement of business processes, Springer-Verlag Berlin Heidelberg, 2011. doi:10.1007/978-3-642-19345-3.

[3] C. Di Ciccio, M. Mecella, On the Discovery of Declarative Control Flows for Artful Processes, ACM Trans. Management Inf. Syst. 5 (4) (2015) 24:1–24:37. doi:10.1145/2629447.

[4] W. M. P. van der Aalst, M. Rosemann, M. Dumas, Deadline-based escalation in process-aware information systems, Decision Support Systems 43 (2) (2007) 492–511. doi:10.1016/j.dss.2006.11.005.

[5] W. M. P. van der Aalst, K. M. van Hee, J. M. E. M. van der Werf, A. Kumar, M. Verdonk, Conceptual model for online auditing, Decision Support Systems 50 (3) (2011) 636–647. doi:10.1016/j.dss.2010.08.014.

[6] M. de Leoni, M. Adams, W. M. P. van der Aalst, A. H. M. ter Hofstede, Visual support for work assignment in process-aware information systems: Framework formalisation and implementation, Decision Support Systems 54 (1) (2012) 345–361. doi:10.1016/j.dss.2012.05.042.

[7] C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling, A. Ruiz-Cortés, RALph: A Graphical Notation for Resource Assignments in Business Processes, in: Int. Conf. on Advanced Information Systems Engineering (CAiSE), Vol. 9097, 2015, pp. 53–68. doi:10.1007/978-3-319-19069-3_4.

[8] W. van der Aalst, H. A. Reijers, M. Song, Discovering Social Networks from Event Logs, Computer Supported Cooperative Work 14 (6) (2005) 549–593. doi:10.1007/s10606-005-9005-9.

[9] M. Song, W. van der Aalst, Towards comprehensive support for organizational mining, Decision Support Systems 46 (1) (2008) 300–317. doi:10.1016/j.dss.2008.07.002.

[10] J. Nakatumba, W. van der Aalst, Analyzing resource behavior using process mining, in: Business Process Management Workshops, 2010, pp. 69–80. doi:10.1007/978-3-642-12186-9_8.

[11] M. Leitner, A. Baumgrass, S. Schefer-Wenzl, S. Rinderle-Ma, M. Strembeck, A Case Study on the Suitability of Process Mining to Produce Current-State RBAC Models, in: Business Process Management Workshops, 2012, pp. 719–724. doi:10.1007/978-3-642-36285-9_72.

[12] A. Baumgrass, M. Strembeck, Bridging the gap between role mining and role engineering via migration guides, Inf. Sec. Techn. Report 17 (4) (2013) 148–172. doi:10.1016/j.istr.2013.03.003.

[13] W. Zhao, X. Zhao, Process Mining from the Organizational Perspective, in: Advances in Intelligent Systems and Computing, Vol. 277, 2014, pp. 701–708. doi:10.1007/978-3-642-54924-3_66.

[14] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, D. Edmond, Workflow Resource Patterns: Identification, Representation and Tool Support, in: Advanced Information Systems Engineering, 2005, pp. 216–232. doi:10.1007/11431855_16.

[15] M. de Leoni, W. M. van der Aalst, M. Dees, A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs, Information Systems 56 (2016) 235–257. doi:10.1016/j.is.2015.07.003.

[16] S. Schönig, C. Cabanillas, S. Jablonski, J. Mendling, Mining the Organisational Perspective in Agile Business Processes, in: Int. Conf. on Enterprise, Business-Process and Information Systems Modeling (BPMDS), Vol. 214 of LNBIP, Springer, 2015, pp. 37–52. doi:10.1007/978-3-319-19237-6_3.

[17] E. Verbeek, J. Buijs, B. van Dongen, W. van der Aalst, XES, xESame, and ProM 6, in: Information Systems Evolution, Vol. 72, 2011, pp. 60–75. doi:10.1007/978-3-642-17722-4_5.

[18] W. van der Aalst, M. Pesic, H. Schonenberg, Declarative workflows: Balancing between flexibility and support, Computer Science - R&D 23 (2) (2009) 99–113. doi:10.1007/s00450-009-0057-9.

[19] OMG, BPMN 2.0, Recommendation, OMG (2011).

[20] M. Zeising, S. Schönig, S. Jablonski, Towards a Common Platform for the Support of Routine and Agile Business Processes, in: IEEE Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing, 2014, pp. 94–103. doi:10.4108/icst.collaboratecom.2014.257269.

[21] M. Montali, Specification and Verification of Declarative open Interaction Models - A logic-based approach, Vol. 56, Springer, 2010. doi:10.1007/978-3-642-14538-4.

[22] F. Maggi, M. Montali, M. Westergaard, W. van der Aalst, Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata, in: Int. Conf. on Business Process Management (BPM), Vol. 6896, Springer, 2011, pp. 132–147. doi:10.1007/978-3-642-23059-2_13.

[23] C. Bussler, Organisationsverwaltung in Workflow-Management-Systemen, Deutscher Universitätsverlag, 1998. doi:10.1007/978-3-663-08832-5.

[24] F. M. Maggi, J. C. Bose, W. van der Aalst, Efficient Discovery of Understandable Declarative Process Models from Event Logs, in: Int. Conf. on

Advanced Information Systems Engineering (CAiSE), Vol. 7328, 2012, pp. 270–285. doi:10.1007/978-3-642-31095-9_18.

[25] F. M. Maggi, A. Mooij, W. van der Aalst, User-Guided Discovery of Declarative Process Models, in: IEEE Symposium on Computational Intelligence and Data Mining, 2011, pp. 192–199. doi:10.1109/CIDM.2011.5949297.

[26] R. J. C. Bose, W. M. van der Aalst, Analysis of Patient Treatment Procedures, in: Business Process Management Workshops, Vol. 99, 2011, pp. 165–166. doi:10.1007/978-3-642-28108-2_17.

[27] F. M. Maggi, J. C. Bose, W. van der Aalst, A Knowledge-Based Integrated Approach for Discovering and Repairing Declare Maps, in: Int. Conf. on Advanced Information Systems Engineering (CAiSE), Vol. 7908, 2013, pp. 433–448. doi:10.1007/978-3-642-38709-8_28.

[28] A. V. Aho, M. R. Garey, J. D. Ullman, The Transitive Reduction of a Directed Graph, SIAM J. Comput. 1 (2) (1972) 131–137. doi:10.1137/0201008.

[29] C. Forgy, Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem, Artif. Intell. 19 (1) (1982) 17–37. doi:10.1016/0004-3702(82)90020-0.

[30] F. M. Maggi, Declarative Process Mining with the Declare Component of ProM, in: Business Process Management Demos, Vol. 1021 of CEUR Workshop Proceedings, 2013.
URL http://ceur-ws.org/Vol-1021/paper_8.pdf

[31] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. Weijters, W. M. van der Aalst, The Need for a Process Mining Evaluation Framework in Research and Practice, in: Business Process Management Workshops, Vol. 4928, 2008, pp. 84–89. doi:10.1007/978-3-540-78238-4_10.

[32] M. Westergaard, C. Stahl, H. Reijers, UnconstrainedMiner: Efficient Discovery of Generalized Declarative Process Models, Tech. Rep. 13-28, Eind-

hoven University of Technology (2013).
URL https://publications.hse.ru/en/preprints/117624631

[33] J. C. Bose, F. M. Maggi, W. van der Aalst, Enhancing Declare Maps Based on Event Correlations, in: Int. Conf. on Business Process Management (BPM), Vol. 8094, 2013, pp. 97–112. doi:10.1007/978-3-642-40176-3_9.

[34] F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi, S. Storari, Exploiting inductive logic programming techniques for declarative process mining, Trans. Petri Nets and Other Models of Concurrency 2 (2009) 278–295. doi:10.1007/978-3-642-00899-3_16.

[35] F. M. Maggi, Discovering Metric Temporal Business Constraints from Event Logs, in: Int. Conf. on Perspectives in Business Informatics Research (BIR), Vol. 194, Springer, 2014, pp. 261–275. doi:10.1007/978-3-319-11370-8_19.

[36] F. M. Maggi, M. Dumas, Discovering Data-Aware Declarative Process Models from Event Logs, in: Int. Conf. on Business Process Management (BPM), Vol. 8094, 2013, pp. 1–16. doi:10.1007/978-3-642-40176-3_8.

[37] S. Rinderle-Ma, W. M. van der Aalst, Life-cycle support for staff assignment rules in process-aware information systems, Tech. rep., Eindhoven University of Technology (2007).
URL http://dbis.eprints.uni-ulm.de/373/

[38] A. Burattin, A. Sperduti, M. Veluscek, Business models enhancement through discovery of roles, in: IEEE Symposium on Computational Intelligence and Data Mining, 2013, pp. 103–110. doi:10.1109/CIDM.2013.6597224.

[39] T. Jin, J. Wang, L. Wen, Organizational modeling from event logs, in: Int. Conf. on Grid and Cooperative Computing (GCC), 2007, pp. 670–675. doi:10.1109/GCC.2007.93.