Mining the Organisational Perspective in Agile Business Processes*

Stefan Schönig[1], Cristina Cabanillas[2], Stefan Jablonski[1], and Jan Mendling[2]

[1]University of Bayreuth, Germany
`{stefan.schoenig,stefan.jablonski}@uni-bayreuth.de`
[2]Vienna University of Economics and Business, Austria
`{cristina.cabanillas,jan.mendling}@wu.ac.at`

Abstract. Agile processes depend on human resources, decisions and expert knowledge, and they are especially versatile and comprise rather complex scenarios. Declarative, i.e., rule-based, process models are well-suited for modelling these processes. Although there are several mining techniques to discover such declarative process models from event logs, they put less emphasis on the organisational perspective, which specifies how resources are involved in the activities. As a consequence, the resulting models do not specify who should execute which task and with which constraint (like separation of duties) in mind. In this paper, we propose a process mining approach to discover resource-aware, declarative process models. Our specific contribution is the extraction of complex rules for resource assignment that integrate the control-flow and organisational perspectives. Our experiments demonstrate the expressiveness of the mined rules with a reference to the Workflow Resource Patterns and a real-world use case.

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

1 Introduction

The success of an organisation primarily depends upon its ability to accomplish its tasks in a structured and reliable manner. A well accepted method for structuring the activities carried out in an organisation is business process management (BPM). BPM usually involves modelling, executing and analysing processes [1]. In this context, two different types of processes can be distinguished [2]: well-structured routine processes with exactly predescribed control flow and agile processes with control flow that evolves at run time without being fully predefined a priori. Agile processes are common in healthcare where, e.g., patient diagnosis and treatment processes require flexibility to cope with unanticipated circumstances.

* This work is funded by the "Europäischer Fonds für regionale Entwicklung" (EFRE) and the Austrian Research Promotion Agency (FFG) under grant 845638 (SHAPE).

In a similar way, two different representational paradigms can be distinguished: procedural models describe which activities can be executed next and declarative models define execution constraints that the process has to satisfy. The more constraints are added to the model, the less possible execution alternatives remain. As agile processes may not be completely known a priori, they can often be captured more easily using a declarative rather than a procedural modelling approach [3–5].

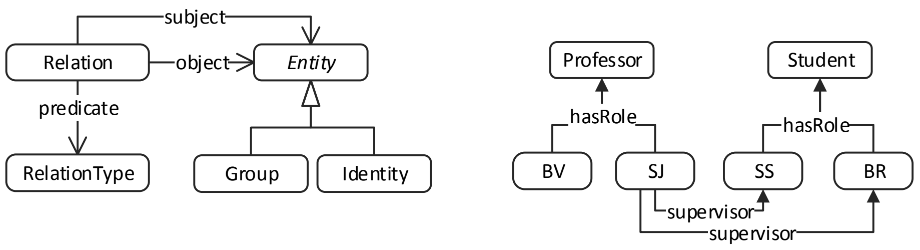
For purposes of compliance and process improvement, organisations are interested in the way their processes are actually executed [1]. Process mining aims at discovering processes by extracting knowledge from event logs, e.g., by generating a process model reflecting the behaviour recorded in the logs [6]. Declarative languages like Declare [3] or DPIL [7] can be used to represent these models, and tools like DeclareMiner [8] or MINERful [9] to generate them, often with a focus on control flow and data [10, 11]. Agile processes, however, need to explicitly integrate the organisational perspective due to the importance of human decision-making and expert knowledge [12]. Recent research has identified the potential of role mining [13, 14] and process mining of the organisational perspective [15]. However, these results have not yet been integrated with declarative process models.

In this paper, we fill this research gap by proposing a process mining approach to discover resource-aware, declarative process models. In particular, we are able to extract complex allocation rules, such as binding of duties between activities, as well as cross-perspective rules involving the control-flow and the organisational perspectives together. The latter consider the influence of resource allocation on the control flow of the process, e.g., an activity can only be executed by a specific role if a specific activity was performed previously. The expressiveness of the extracted rules has been evaluated using the Workflow Resource Patterns [16]. The applicability of the approach has been validated with a real-world event log from the university domain.

The remainder of this paper is structured as follows: In Section 2 we explain background upon which our approach is built. In Section 3 we describe our approach to extract resource-aware, declarative process models. In Section 4 we provide details about the implementation as well as experimental results. In Section 5 we present the results of the evaluations performed. In Section 6 we describe the related work and Section 7 concludes this paper.

2 Background

This section describes the language that we will use as a basis for mining models. We choose *Declarative Process Intermediate Language (DPIL)* [7] for this purpose due to several reasons. First, it is multi-perspective, i.e., it allows representing several business process perspectives, namely, control flow, data and resources. Since we want to extract resource-aware process models, the modelling language needs to support the modelling of rules related to the organisational perspective. The expressiveness of DPIL and its suitability for business pro-



(a) Organisational meta model (b) Organisational model

Fig. 1: Organisational meta model and example organisational model

cess modelling have been evaluated [7] with respect to the well-known Workflow Patterns [16]. Second, it is multi-modal, meaning that it allows defining two different types of rules: rules representing mandatory relations (called *ensure* in DPIL) and rules representing recommended relations (called *advice* in DPIL). The latter are useful, e.g., to reflect good practices.

In order to express organisational relations, DPIL builds upon a generic organisational meta model that has been described in [17] and is depicted in Fig. 1a. 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 suitable to define, 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. Fig. 1b illustrates an exemplary organisational model of a university research group. It is composed of two roles (Professor, Student) assigned to four people (BV, SJ, SS, BR) and several 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* macro ($sequence(a,b)$) 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* macro ($role(a,r)$) states that an activity a is assigned to a role r. Fig. 2 shows an example of a process for trip management modelled with DPIL that uses the organisational model defined in Fig. 1b. It states that it is mandatory to approve a business trip before a flight can be booked. Moreover, it is recommended but not necessary that the approval be carried out by a resource with the role *Professor*.

In the work at hand, we use DPIL and build upon the described organisational meta model. However, please note that any other declarative process modelling language like Declare [3] or Dynamic Condition Response graphs [18] in combination with a suitable organisational meta model that fulfils the identified requirements can be used.

```
use group Professor

process BusinessTrip {
   task Book flight
   task Approve Application

   advice role(Approve Application, Professor)
   ensure sequence(Approve Application, Book flight)
}
```

Fig. 2: Process for trip management modelled with DPIL

Applying process mining techniques it is possible to generate process models like the one depicted in Fig. 2, as long as the event logs contain the required information. Each event in a log refers to an activity, i.e., a well-defined step in the process, and is related to a particular process instance. We refer to the ordered set of events of a particular process instance as a *trace*. Event logs usually contain information about the resource performing an activity [6], as well as additional information that may be useful for subsequent analysis purposes. For instance, the following excerpt of a business trip process event log encoded in the XES logging format [19] shows the recorded information of the *start event* of an activity *Apply for trip* performed by a resource *SS*.

```
<string key="concept:name" value="SS_Riga2013"/>
<event>
   <string key="org:resource" value="SS"/>
   <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>
```

3 Mining Resource-Aware Declarative Process Models

In this section, we describe our approach to automatically discover resource-aware, multi-modal, declarative process models from event logs. First, we describe rule candidates and a support and confidence framework. Finally, we cover rule templates along with an improvement based on pre- and post-processing.

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 with macros in textual languages (e.g., in DPIL) that make the model easier to understand. In contrast to concrete rules, a rule template consists of placeholders, i.e., typed variables. It 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 and to find solutions to the placeholders. A solution to the query 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. The *sequence* template, e.g., consists of 2 placeholders of type $Task$. Assuming that $|T|$ different tasks occur in the event log, $|T|^2$ *rule candidates* are generated. All the resulting rule candidates are subsequently checked w.r.t. the event log.

In many cases, a rule candidate can be trivially valid. Consider, e.g., the rule 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 an identity i_1, and the example event log of Table 1. The provided event log notation (first column) encodes the recorded 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 [20], we say that the rule is vacuously satisfied. It is necessary to discriminate between traces where a rule is trivially true and traces in which the rule is non-vacuously satisfied. Only traces in which a rule candidate non-trivially holds are considered interesting [8]. For first order logic rules that depict implications of the form $A \rightarrow B$ like in DPIL, trivially and non-vacuously valid rules can be discriminated by additionally checking the condition A of the rule separately.

Table 1 also shows the results of checking the non-vacuous satisfaction of the direct(t_1,i_1) rule (third column) as well as its condition (second column) for each trace of the example event 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 while it is violated in trace five.

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 exemplary rule and its condition

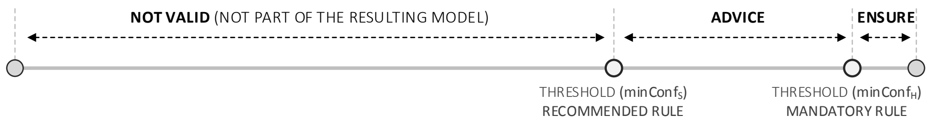


Fig. 3: Classification of rule candidates based on the confidence value

3.2 Support and Confidence Framework to Classify Rules

Checking rule candidates as described above provides for every candidate the number of instances, i.e., 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. Therefore, [20] adopted a support and confidence framework proposed by association rule mining methods to evaluate the relevance of rule candidates.

Definition 1 (Support and Confidence). *Let $|\Phi|$ be the 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 $supp(r)$ and confidence $conf(r)$*[1] *values of a rule r are defined as:*

$$supp(r) := \frac{|\sigma_{nv}(r)|}{|\Phi|}, \qquad conf(r) := \frac{supp(r)}{supp(A)} \tag{1}$$

Considering the event log of Table 1 and the direct(t_1,i_1) rule, its support evaluates to $supp(r) = 0.6$ and its confidence to $conf(r) = 0.75$. The support value is used for pre-processing, as described in Section 3.4. We make use of the confidence value in order to classify a rule candidate r as (i) a *mandatory* rule, i.e., satisfied in almost all traces; (ii) a *recommended* rule, i.e., not always satisfied but with a tendency to be satisfied; or (iii) a *non-valid* rule, i.e., violated in most of the recorded traces. As visualized in Fig. 3, two thresholds $minConf_S$ and $minConf_H$ are introduced to classify rule candidates. Candidates r with $conf(r) > minConf_H$ are classified as mandatory (*ensure*) and $minConf_S < conf(r) < minConf_H$ as recommended (*advice*). All rule candidates r with $conf(r) < minConf_S$ are non-valid rules and are not part of the resulting process model. Using the confidence values of rule candidates it is directly possible to generate a DPIL process model reflecting the recorded behaviour.

3.3 Rule Templates for Analysing the Organisational Perspective

The previous section showed how it is possible to automatically generate a multi modal, declarative process model by checking a set of rule candidates whose structure is defined by rule templates w.r.t. a given event log. Since DPIL builds upon a flexible organisational meta model, it is possible to define rule

[1] In case of rules that do not depict implications, the condition is satisfied in every trace. Here $supp(A) = 1$ and $conf(r) = supp(r)$.

templates that define the structure of organisational relations. By instantiating these resource-aware 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 corresponding organisational model. Based on the resulting confidence values, a resource-aware process model can be generated automatically.

We now define rule templates and their macros that can be used to mine the organisational perspective and to generate a resource-aware, declarative process model. We identified three different groups of organisational rule templates: resource allocation templates related to a single task, resource allocation templates related to more than one task, and cross-perspective rule templates, i.e., templates that express the influence of resources on the execution order of tasks. For every group we provide at least two representative examples that cover frequently needed organisational relations, according to the Workflow Resource Patterns [16]. Nonetheless, further rule templates can be defined individually according to the analyst's needs.

We first focus on rule templates that define *resource allocation* patterns, i.e., rules that specify the resources which are allowed to perform a certain task. The *direct allocation* of resources to a task can be extracted by analysing the *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)
```

Role-based allocation of resources can be identified with the *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)
```

Organisational patterns can also relate to more than one task at the same time. The *binding(*T_1*,*T_2*)* template, e.g., can be used to discover if a task is always (mandatory) or should (recommended) be performed by the same resource as another task. With the *separate(*T_1*,*T_2*)* template, on the contrary, it is possible to discover task combinations that need to be or should be performed by different resources. For both templates, $|T|^2$ candidates need to be checked.

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

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

The *orgDist(*T_1*,*T_2*,RT)* template is used to discover relations that are defined in the organisational model between the resources that performed two different tasks. This template incorporates a variable RT for the different relation types in the considered organisational model. Like this, $|T|^2 \cdot |RT|$ rule candidates exist.

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

Moreover, the organisational perspective can affect the execution order of tasks, i.e., the control flow of the process. There may be processes in which a sequence between tasks only holds for specific resources or for resources with a specific role. These patterns can be discovered by the *roleSequence(T_1,T_2,G)* and the *resourceSequence(T_1,T_2,I)* templates, respectively. For these templates, $|T|^2 \cdot |G|$ and $|T|^2 \cdot |I|$ candidates need to be checked.

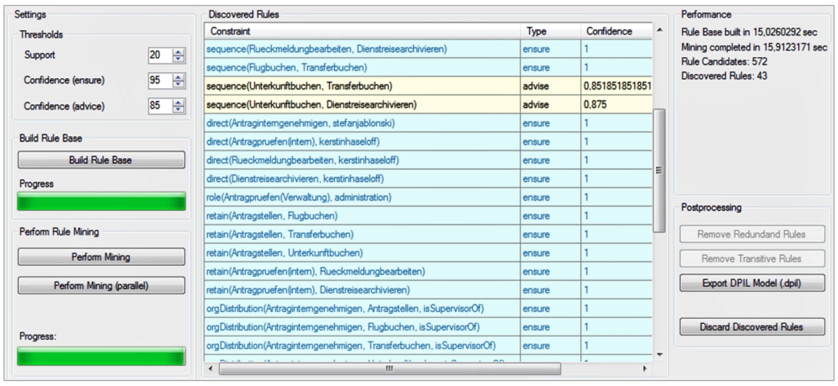
```
roleSequence(T1,T2,G) iff start(of T2 by :p at :t) and
                      relation(subject p predicate hasRole object G)
                      implies complete(of T1 at < t)

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

3.4 Pre- and Post-processing

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 [21] of a hospital information system 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 also becomes obvious when considering task/resource combinations of the event log in Table 1. The 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 by Maggi et al. [20] uses the well-known Apriori algorithm to pre-process the log and to extract *task combinations* that frequently occur together. A task combination is considered to be relevant if it occurs in a sufficient number of traces, i.e., above a given threshold *minSupp*. A *minSupp* of 5%, e.g., claims that only rule candidates are considered whose parameter combinations occur in at least 5% of the recorded traces. We extended this method in [22] to also extract *task/resource* and *task/group* combinations that frequently occur together. This way, it is also possible to dramatically reduce the number of organisational rule candidates by abstracting from infrequent parameter combinations. Hence, for the example log, only 1 out of 3 *direct(T,i_4)* candidates are generated and checked.

Furthermore, when automatically generating a declarative process model, there are potentially extracted rules that are redundant. Consider, e.g., that a specific task t_1 has always been performed by a resource i_1 who has a role g_1 according to the organisational model. Then, the proposed method will (inevitably) discover a *role(t_1,g_1)* rule. This rule is redundant, since a direct allocation rule *direct(t_1,i_1)* will also be discovered. In case of i_1 *hasRole* g_1 the *role* rule is already implied in the *direct* rule. Redundant rules complicate the



Support	20
Confidence (ensure)	95
Confidence (advice)	85

Constraint	Type	Confidence
sequence(Rueckmeldungbearbeiten, Dienstreisearchivieren)	ensure	1
sequence(Flugbuchen, Transferbuchen)	ensure	1
sequence(Unterkunftbuchen, Transferbuchen)	advise	0,851851851851
sequence(Unterkunftbuchen, Dienstreisearchivieren)	advise	0,875
direct(Antragintemgenehmigen, stefanjablonski)	ensure	1
direct(Antragpruefen(intern), kerstinhaseloff)	ensure	1
direct(Rueckmeldungbearbeiten, kerstinhaseloff)	ensure	1
direct(Dienstreisearchivieren, kerstinhaseloff)	ensure	1
role(Antragpruefen(Verwaltung), administration)	ensure	1
retain(Antragstellen, Flugbuchen)	ensure	1
retain(Antragstellen, Transferbuchen)	ensure	1
retain(Antragstellen, Unterkunftbuchen)	ensure	1
retain(Antragpruefen(intern), Rueckmeldungbearbeiten)	ensure	1
retain(Antragpruefen(intern), Dienstreisearchivieren)	ensure	1
orgDistribution(Antragintemgenehmigen, Antragstellen, isSupervisorOf)	ensure	1
orgDistribution(Antragintemgenehmigen, Flugbuchen, isSupervisorOf)	ensure	1
orgDistribution(Antragintemgenehmigen, Transferbuchen, isSupervisorOf)	ensure	1

Performance
Rule Base built in 15,0260292 sec
Mining completed in 15,9123171 sec
Rule Candidates: 572
Discovered Rules: 43

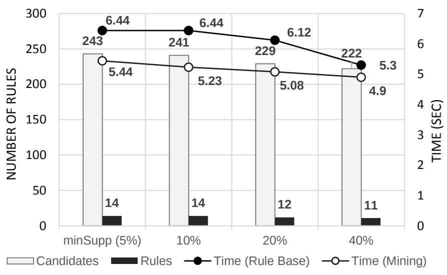
Fig. 4: User-interface of the DpilMiner during the analysis of an event log

understandability of discovered models. Maggi et al. [23] proposed a technique to post-process a discovered model and to remove redundant, *weaker* rules if they are already implied in *stronger* rules only focusing on the hierarchy of control flow perspective templates. We extended this method to also consider the rule hierarchies of organisational rules. Redundancy may also be caused by the interplay of three or more organisational rules. Consider, e.g., 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 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. Not all rule types can be reduced using transitive reduction. *Separate* rules, e.g., 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 automatically that t_1 is also not performed by the same resource as t_3.

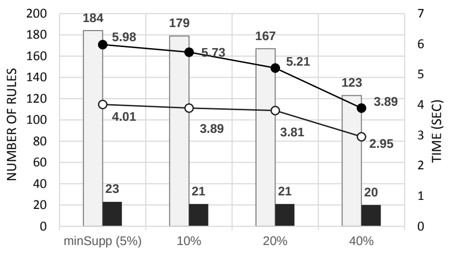
4 Implementation and Experiments

The problem of checking a large set of rule candidates can be solved by efficient pattern matching methods like the *rete algorithm* [24]. 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 [25] provides a current implementation of this method. In order to check rule candidates with Drools, they are translated to the Drools Rule Language (DRL).

The described approach has been implemented in the *DpilMiner* application. Fig. 4 shows the DpilMiner user-interface and some discovered rules when analysing the application to an event log. To analyse performance and applicability, we applied the DpilMiner with different configurations using an event log



(a) Results using template set 1



(b) Results using template set 2

Fig. 5: Analysis of the mining approach with different template sets

of a university business trip management system[2] The log contains 2104 events of 10 different activities related to the application and the approval of university business trips as well as the management of accommodations and transfers, i.e., booking hotels, flights or trains. The system has been used for 6 months by 11 employees. In order to show the complete functionality of our approach, the underlying organisational model of the domain is required. The given organisational model assigns these identities to 4 distinct roles, specifically, 6 phd students, 2 professors, 1 secretary as well as 2 administration employees. In total, there are 128 business trip cases recorded in the log. 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. 5 outlines the results. Fig. 5a shows the results of applying the approach with template *set* 1, which contains four different rule templates purely focusing on the organisational perspective, i.e., *direct*, *role*, *binding* as well as the *orgDist* template. The diagram in Fig. 5b shows the results for template *set* 2, which analyses the execution order of tasks under consideration of the organisational perspective, i.e., the templates *sequence* and *roleSequence*. We analysed the time to build the rete network, i.e., the rule base, as well as the time to perform the actual mining process under consideration of a different number of rule candidates. This was achieved by considering different $minSupp$ values during the preprocessing phase. 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 of *set* 2 additionally highlights the benefit of the pre-processing approach. With increasing $minSupp$, the number of rule candidates to check considerably decreases. However, almost the same number of rules has been discovered.

In order to evaluate and compare the performance of our approach, we also applied the ProM implementation of the DeclareMiner [26] by only analysing the *precedence* template of Declare [3], 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

[2] The event log is available for download at *workbench.kppq.de*.

	Resource Pattern	**DPIL Rule Template**
✓	Direct Allocation	start(of T) implies start(of T by :p)
✓⋆	Role-Based Allocation	start(of T by :p) implies relation(subject p predicate hasRole object G)
⊘	Deferred Allocation	—
⊘	Authorization	—
✓	Separation of Duties	start(of T_1 by :p) and start(of T_2) implies start(of T_2 by not p)
✓	Case Handling	forall(task A start(of T) implies start(of T by :p)
✓	Retain Familiar	start(of T_1 by :p) and start(of T_2) implies start(of T_2 by p)
✓⋆	Capability-Based Allocation	start(of T by :p) implies relation(subject p predicate RT object G)
⊘	History-Based Allocation	—
✓⋆	Organisational Allocation	start(of T_1 by :p_1) and start(of T_2 by :p_2) implies relation(subject p_1 predicate RT object p_2)
⋆	Automatic Execution	invoke(T)

✓=Supported ✓⋆=Organisational model required ⊘=Non supported

Table 2: Overview of Workflow Resource Patterns that can be discovered

example log with 2 respectively 4 rule templates, our approach was still faster in any case, as depicted in Fig. 5. Note that the rule base only needs to be built once for several different mining applications.

5 Evaluation

In the following section, we evaluate the expressiveness and the applicability of the described approach.

5.1 Discovering Creation Workflow Resource Patterns

We use the group of so-called *creation patterns* of the well-known Workflow Resource Patterns [16] to evaluate which of the resource allocation patterns can be discovered by our mining approach. The Workflow Resource Patterns have been used in several modelling approaches as a reference model to specify how resources take part in process activities. As described in Section 3, our approach is able to discover a particular resource pattern if it is possible to define a parametrized DPIL rule template that represents the pattern.

Table 2 shows an overview of the different resource patterns as well as the corresponding rule templates, when possible. As seen, the patterns *Direct Allocation*, *Role-Based Allocation*, *Separation of Duties*, *Retain Familiar* and *Organisational Allocation* are extractable with the rule templates *direct*, *role*, *separate*,

```
advice direct(Approve Application, SJ)
ensure role(Approve Application, Professor)
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)
advice orgDist(Approve Application, Apply for trip, supervisor)
advice sequence(Apply for trip, Book flight)
ensure roleSequence(Apply for trip, Book flight, Student)
```

Fig. 6: Excerpt of the discovered business trip process model

binding and *orgDist*, respectively. Since event logs do not usually contain information about the resource assignment mechanism used, it is not possible to extract if the resource assignment has been deferred to run time, i.e., the *Deferred Allocation* pattern cannot be discovered. Since in DPIL it is not possible to specify relations that relate to other process instances, the *History-Based Allocation* pattern cannot be discovered. The *Automatic Execution* pattern can only be discovered if the event log makes use of certain event types which indicate an automatic task processing. DPIL, e.g., defines invoke(T) events to call an automatic non-human service T. Even without considering invoke events our approach is able to discover 7 out of 11 resource creation patterns.

5.2 Case Study

In this section, we describe our findings when applying the approach to the university business trip event log that has already been described in Section 4. We analysed the event log with 6 different organisational rule templates comprising resource allocation patterns as well as resource influence on task execution order, i.e., cross-perspective patterns. With $minSupp = 10\%$ in the pre-processing phase and after removing redundant rules in the post-processing phase, we extracted 35 rules in total. The extracted resource allocation patterns are composed of 4 *direct*, 1 *role*, 5 *retain* and 4 *orgDist* rules. The control flow related pattern set is composed of 13 *sequence* and 8 *roleSequence* rules. For the classification in mandatory and recommended rules, we used $minConf_H = 95\%$ and $minConf_S = 85\%$. For space reasons, we only describe some interesting parts of the resulting model in Fig. 6.

We first focus on interesting resource allocation patterns. The discovered model shows that tasks "Approve Application" and "Check Application" have *always* been performed by a resource with the role "Professor" and "Administration", respectively (mandatory role-based allocation). Even if different professors performed the approval, the execution by the identity "SJ" is preferred (recommended direct allocation). The three binding of duties rules show that the resource who booked the flight, the accommodation as well as the transfer service has to be the applicant herself (mandatory binding of duties). The model additionally shows that the resource who approved the trip application tends to be the supervisor of the applicant (recommended organisational distribution). Moreover, we also discovered the influence of resources on the execution order of tasks. Although employees usually applied for the trip before they booked the corresponding flight, it is not mandatory (recommended task sequence). Nonetheless, there are cases in which certain employees already booked the flight without applying for the trip. However, when analysing the ordering of tasks under consideration of performing resources, we extracted that students always applied for the trip before they booked a flight (mandatory role-based sequence). While professors are free to book a flight without an approved application, students mandatorily have to stick to a certain order of tasks.

In order to evaluate the precision of the mining results, the model has finally been discussed in a workshop with process particpants. Here, 30 out of 35 rules have been identified as relevant while 5 rules have been classified as non-relevant. This leads to a precision value of 0.86.

6 Related Work

The work presented in this paper relates to two streams of research: declarative versus procedural process mining and mining of the organisational perspective of a process. Recently, several techniques for the automated discovery of declarative process models from event logs have been proposed. The DeclareMiner [8] and its enhancements aim to improve the mining performance [20] as well as the readability of discovered models [10, 11, 23]. Furthermore, efficient algorithms to discover Declare models are presented in [9, 27, 28]. The work on Dynamic Condition Response Graphs [18] proposes an alternative formalism. In essence, the focus of these approaches is control flow with extensions to cover data. Complementary to these papers are approaches on role mining [13, 14] and process mining of the organisational perspective [15] that aim to make use of the rich information on who has been executing a particular task [11]. Mining methods for analysing event logs with respect to resource information are mainly focused on enriching a given procedural model with resource information [15], on extracting an underlying organisational model [29] or social network [30], or on analysing the influence of resources on process performance [31]. Approaches on role mining [13, 14] are interested in separation of duty constraints. The research reported in this paper takes a step towards the integration of both streams and the mining of constraints that express resource assignments depending on control flow and vice versa. In this way, it has the potential for evolving to a useful tool for compliance management of agile processes.

7 Conclusions and Future Work

In this paper, we presented a process mining method to discover resource-aware and multi-modal, declarative process models. Our approach is based upon the textual Declarative Process Intermediate Language (DPIL), in which organisational relations of processes can be modelled. We proposed a set of rule templates that can be used to mine the organisational perspective and to generate a resource-aware declarative model. Our approach has been implemented in the DpilMiner application. We analysed performance and tested its applicability using an event log of a university business trip management system. Moreover, we evaluated the expressiveness against the Workflow Resource Patterns.

Since our approach is based upon DPIL, the mining capabilities are limited to the expressiveness of the language. Hence, e.g., inter-case dependencies such as those in the History-Based Allocation pattern cannot be discovered. Furthermore, the analysis of certain rule templates leads to many discovered rules. The *separate* template, e.g., discovers all combinations of tasks, which are performed by different resources. It is in some cases difficult to manage the amount of rules. Hence, mechanisms to reduce the rule set must be explored. Finally, we are currently evaluating possibilities to map the DPIL to existing graphical process modelling notations, such as RALph [32]. This will increase the understandability of the resulting process models for systems analysts.

References

1. M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management.* Springer, 2013.
2. S. Jablonski, "MOBILE: A modular workflow model and architecture," in *Working Conference on Dynamic Modelling and Information Systems*, 1994.
3. W. van der Aalst, M. Pesic, and H. Schonenberg, "Declarative workflows: Balancing between flexibility and support," *Computer Science - Research and Development*, vol. 23, no. 2, pp. 99–113, 2009.
4. P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, and H. Reijers, "Imperative versus declarative process modeling languages: An empirical investigation," *Business Process Management Workshops*, pp. 383–394, 2012.
5. R. Vaculín, R. Hull, T. Heath, C. Cochran, A. Nigam, and P. Sukaviriya, "Declarative business artifact centric modeling of decision and knowledge intensive business processes," in *EDOC*, pp. 151–160, 2011.
6. W. van der Aalst, *Process mining: discovery, conformance and enhancement of business processes.* 2011.
7. M. Zeising, S. Schönig, and S. Jablonski, "Towards a Common Platform for the Support of Routine and Agile Business Processes," in *Collaborative Computing: Networking, Applications and Worksharing*, 2014.
8. F. M. Maggi, A. Mooij, and W. van der Aalst, "User-Guided Discovery of Declarative Process Models," in *Computational Intelligence and Data Mining*, pp. 192–199, 2011.
9. C. Di Ciccio and M. Mecella, "A two-step fast algorithm for the automated discovery of declarative workflows," in *Computational Intelligence and Data Mining*, pp. 135–142, 2013.
10. F. M. Maggi and M. Dumas, "Discovering Data-Aware Declarative Process Models from Event Logs," in *Business Process Management*, pp. 1–16, 2013.
11. J. C. Bose, F. M. Maggi, and W. van der Aalst, "Enhancing Declare Maps Based on Event Correlations," in *Business Process Management*, pp. 97–112, 2013.

12. M. Marin, R. Hull, and R. Vaculín, "Data Centric BPM and the Emerging Case Management Standard : A Short Survey Case Management," vol. 257593.
13. M. Leitner, A. Baumgrass, S. Schefer-Wenzl, S. Rinderle-Ma, and M. Strembeck, "A case study on the suitability of process mining to produce current-state rbac models," in *Business Process Management Workshops*, pp. 719–724, 2013.
14. A. Baumgrass and M. Strembeck, "Bridging the gap between role mining and role engineering via migration guides," *Inf. Sec. Techn. Report*, vol. 17, no. 4, pp. 148–172, 2013.
15. W. Zhao and X. Zhao, "Process Mining from the Organizational Perspective," in *Foundations of Intelligent Systems*, pp. 701–708, 2014.
16. N. Russell, W. M. van der Aalst, A. H. Ter Hofstede, and D. Edmond, "Workflow resource patterns: Identification, representation and tool support," in *Advanced Information Systems Engineering*, pp. 216–232, 2005.
17. C. Bussler, *Organisationsverwaltung in Workflow-Management-Systemen.* Dt. Univ.-Verlag, 1998.
18. T. T. Hildebrandt, R. R. Mukkamala, T. Slaats, and F. Zanitti, "Contracts for cross-organizational workflows as timed dynamic condition response graphs," *J. Log. Algebr. Program.*, vol. 82, no. 5-7, pp. 164–185, 2013.
19. E. Verbeek, J. Buijs, B. van Dongen, and W. van der Aalst, "XES, xESame, and ProM 6," in *Information Systems Evolution*, pp. 60–75, 2011.
20. F. M. Maggi, J. C. Bose, and W. van der Aalst, "Efficient Discovery of Understandable Declarative Process Models from Event Logs," in *Advanced Information Systems Engineering*, pp. 270–285, 2012.
21. R. J. C. Bose and W. M. van der Aalst, "Analysis of Patient Treatment Procedures," in *Business Process Management Workshops*, pp. 165–166, 2011.
22. S. Schönig, F. Gillitzer, M. Zeising, and S. Jablonski, "Supporting rule-based process mining by user-guided discovery of resource-aware frequent patterns," in *ICSOC 2014 Workshops, in press*, 2014.
23. F. M. Maggi, J. C. Bose, and W. van der Aalst, "A Knowledge-Based Integrated Approach for Discovering and Repairing Declare Maps," in *Advanced Information Systems Engineering*, pp. 433–448, 2013.
24. C. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial intelligence*, vol. 19, no. 1, pp. 17–37, 1982.
25. T. JBoss Drools, "JBoss Drools Documentation - Chapter 7: Rule Language Reference," 2013.
26. F. M. Maggi, "Declarative Process Mining with the Declare Component of ProM," in *BPM (Demos)*, 2013.
27. M. Westergaard, C. Stahl, and H. Reijers, "UnconstrainedMiner: Efficient Discovery of Generalized Declarative Process Models," *BPM Center Report, No. BPM-13-28*, 2013.
28. C. Di Ciccio, F. M. Maggi, and J. Mendling, "Discovering target-branched declare constraints," in *Business Process Management*, pp. 34–50, 2014.
29. M. Song and W. van der Aalst, "Towards comprehensive support for organizational mining," *Decision Support Systems*, 2008.
30. W. M. Van Der Aalst, H. A. Reijers, and M. Song, "Discovering Social Networks from Event Logs," *CSCW*, vol. 14, no. 6, pp. 549–593, 2005.
31. J. Nakatumba and W. van der Aalst, "Analyzing resource behavior using process mining," in *Business Process Management Workshops*, pp. 69–80, 2010.
32. C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling, and A. Ruiz-Cortés, "RALph: A Graphical Notation for Resource Assignments in Business Processes," in *CAiSE*, p. In press, 2015.