Stefan Schönig and Cristina Cabanillas Macias and Claudio Di Ciccio and Stefan Jablonski and Jan Mendling

Mining team compositions for collaborative work in business processes

Article (Accepted for Publication)
(Refereed)

Noname manuscript No.
(will be inserted by the editor)

Mining Team Compositions for Collaborative Work in Business Processes

Stefan Schönig · Cristina Cabanillas · Claudio Di Ciccio · Stefan Jablonski · Jan Mendling

Received: date / Accepted: date

Abstract Process mining aims at discovering processes by extracting knowledge about their different perspectives from event logs. The resource perspective (or organisational perspective) deals, among others, with the assignment of resources to process activities. Mining in relation to this perspective aims to extract rules on resource assignments for the process activities. Prior research in this area is limited by the assumption that only one resource is responsible for each process activity and hence, collaborative activities are disregarded. In this paper, we leverage this assumption by developing a process mining

Stefan Schönig
Vienna University of Economics and Business, Austria
Tel.: +43-1-313365216
Fax: +43-1-31336905216
E-mail: stefan.schoenig@wu.ac.at

Cristina Cabanillas
Vienna University of Economics and Business, Austria
Tel.: +43-1-313365216
Fax: +43-1-31336905216
E-mail: cristina.cabanillas@wu.ac.at

Claudio Di Ciccio
Vienna University of Economics and Business, Austria
Tel.: +43-1-313365222
Fax: +43-1-31336905222
E-mail: claudio.di.ciccio@wu.ac.at

Stefan Jablonski
University of Bayreuth, Germany
Tel.: +49 (0)921557620
Fax: +49 (0)921557622
E-mail: stefan.jablonski@uni-bayreuth.de

Jan Mendling
Vienna University of Economics and Business, Austria
Tel.: +43-1-313365200
Fax: +43-1-31336905200
E-mail: jan.mendling@wu.ac.at

approach that is able to discover team compositions for collaborative process activities from event logs. We evaluate our novel mining approach in terms of computational performance and practical applicability.

Keywords Business process management · declarative process mining · event log analysis · resource perspective · teamwork

1 Introduction

Business process management is a well accepted method for structuring the activities carried out in an organisation in order to achieve improvements in terms of efficiency and effectiveness [15]. Processes are not always explicitly defined, which raises the need to discover the implicit rules according to which they are executed. Process mining provides different methods, a.o., for automatically discovering processes by extracting knowledge from event logs, e.g., by generating a process model reflecting the behaviour recorded in the logs. Various algorithms are available to discover models capturing the control flow [1] and the organisational perspective [43] of a business process. The latter focuses on the assignment of resources to activities of the process.

While the scope of the approaches that focus on control flow aspects is very broad, the mining of resources is usually restricted by the assumption that individual activities are performed by exactly one person. However, in domains like healthcare, software development, and knowledge-intensive processes in general, most of the activities are carried out collaboratively, such that several human resources are involved with working on a single activity [12]. Such resources work as a team in which each team member has a set of capabilities that generally make them play a specific role in the team. These characteristics are the ones being matched to the task requirements in order to compose a team suitable to work on it [22]. There is usually a hierarchical relation among the members of the team, though bigger teams could be organised according to more complex structures, such us holarchies or federations. Flat teams in which no organisational relation exists between team members are called coalitions. When teams are long-lived and serve several purposes they are called congregations [22]. We will use the term *team* indistinctly in this paper. Domains in which collaborative work is frequent can greatly benefit from approaches towards mining team compositions, which unveil the capabilities and organisational relations of the team members. Such approaches for mining collaborative work in business processes are currently missing.

In this paper, we address this research gap by extending a declarative process mining framework [35] towards the integration of collaborative activities. The approach consists of a two-step mining approach that first extracts the teams participating in a collaborative activity from an event log and then discovers the overall characteristics of the team members in terms of the skills, organisational roles, etc., that are present in the team; afterwards, a two-step post-processing phase derives the most informative team compositions including the distribution of the discovered characteristics among the team

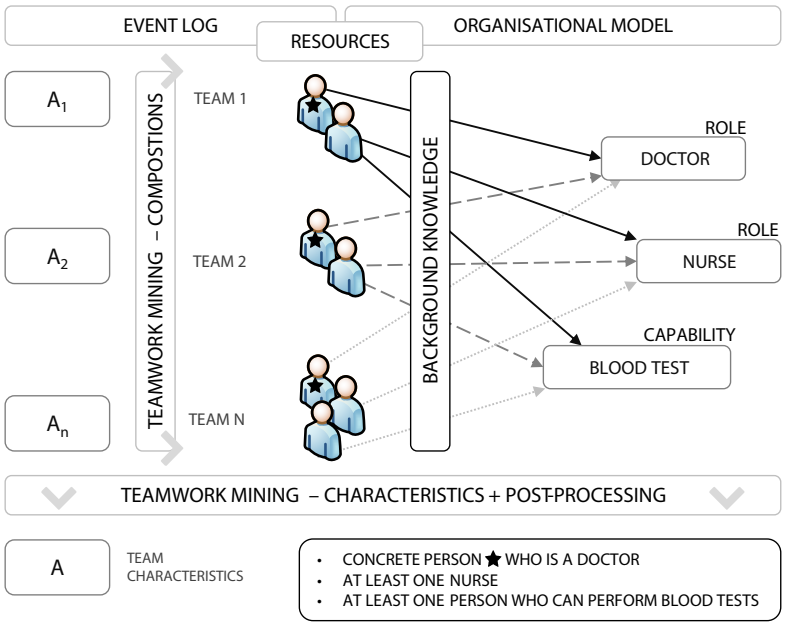


Fig. 1: Approach for mining team compositions

members. An example is depicted in Fig. 1. From the first mining step three different teams are extracted from an event log for a collaborative activity A. After analysing the teams against the existing background knowledge from the organisational model, the team characteristics *role Doctor*, *role Nurse* and *capability Blood Test* are discovered for these teams. Finally, the post-processing phase concludes that the teams performing that activity must always include a specific person $\star$ who, in turn, is a doctor; at least one nurse; and at least one person that can perform blood tests.

Our approach has been evaluated in three directions. First, we have implemented the approach in a working prototype. Furthermore, we have tested its practical applicability using real-life event logs from two different domains. Additionally, we have measured the computational performance by conducting experiments. With this work we extend the scope of process mining to a larger number of processes and hence, to a larger set of domains, including individual and collaborative tasks.

This article differs from the conference publication [35] in the following aspects: *(i)* the original mining approach has generally been enhanced and adapted to analyse resource assignment and team compositions of *collaborative* activities; *(ii)* a two-step post-processing phase for deriving the most informative team compositions has been developed; *(iii)* the implementation of every mining module is described in detail, and *(iv)* the approach has been evaluated w.r.t. two real-life event logs from different domains.

The remainder of the paper is structured as follows: Section 2 describes the background that constitutes the starting point for developing the approach. Section 3 describes the teamwork mining approach. Section 4 explains the evaluations performed. Finally, Section 5 concludes the paper and discusses directions for future work.

2 Background

Prior research related to our goals deals with mining the organisational perspective of business processes, providing support for collaboration in workflow systems and modelling organisational information in the context of a business process.

There are already approaches for mining the organisational perspective of business processes [46] that complement traditional methods for mining process control flow. Some approaches analyse the influence of resources on process performance [30]. Other approaches extract organisational entities (e.g., roles) and static relations between process participants (e.g., delegation) to discover the underlying organisational model [37] or social network [40]. Advanced research has been capable of extracting also dynamic relations regarding the actual communication between the process participants using enriched event logs [21].

Most of the existing mining methods, however, are focused on enriching a given procedural model with resource information [46]. These methods make use of the rich information available on who has executed a particular task in past process executions [8] to discover the characteristics of the participants of a given task. They differ in the way and degree to which they extract certain types of assignment rules. The workflow resource patterns, and specifically the creation patterns, provide a basis for discussing these differences [33]. Table 1 shows the patterns supported by the different approaches. The approaches that are capable of mining a Role-Based Access Control (RBAC) model focus on discovering role-based resource assignments (a.k.a. role mining) [23] and corresponding access-control constraints (e.g., separation and binding of duties) [6,26,37]. Staff assignment mining [27] deals with extracting complex assignment rules based on the capabilities of a resource and organisational hierarchy using decision tree learning. Here, resource assignment can only be identified in relation to a single task – rules referring to several tasks, such as separation of duties, cannot be discovered. Mining approaches that use more generic organisational metamodels are capable of discovering most of the creation patterns as well as cross-perspective patterns that consider the control flow and the organisational perspectives altogether [35]. It has to be noted, though, that all the existing approaches assume individual activities to be performed by a single resource and disregard collaborative work. They do not provide the means to directly discover team composition within a business process.

Resource Pattern	**Mining approach**
Direct Distribution	[37,6,27,23,35]
Role-based Distribution	[37,6,27,23,35]
Deferred Distribution	-
Separation of Duties	[26,6,35]
Case Handling	[37,6,35]
Retain Familiar	[26,6,35]
Capability-based Distribution	[27,35]
History-based Distribution	-
Organisational Distribution	[27] (single task), [35]
Cross-Perspective Patterns	[35]

Table 1: Existing approaches for mining organisational information

The necessity of dealing with individual and collaborative activities in the same modelling (and operating) environment has been identified and some partial solutions have been proposed in the domain of computer supported collaborative work [31,20]. A survey on teamwork over the past fifty years identified a set of challenges [34] from several perspectives. Relevant to our research are the approaches dealing with team composition and team modelling, since they may provide hints on the expected outcome of a teamwork mining approach.

Most of the approaches that deal with team composition and selection address the problem of finding the best match of experts to required skills [16,18,7,5]. Several of these approaches study connectivity and social aspects for team composition, e.g., social distance between people [44]. Dorn et al. [13] highlight physical location and communication capabilities between team members as relevant. They present an approach for deriving user profiles from social networks and create virtual teams in which there is balance between skills and connectivity. This is extended towards a skill-dependent recommendation model for team composition [14]. Some other approaches considering both skills and connectivity are described in [25,36,11].

As far as team modelling is concerned, several organisational metamodels considering teamwork have been proposed. STEAM [38] defines an organisational metamodel to support hierarchies of teams, composed of individuals. Both teams and people can be associated to roles according to their capabilities. Roles can be persistent or task-specific. Tambe et al. [39,24] investigate how that metamodel performs in building agent-teams in the simulation league for Robocup, and how agents learn specific skills. Van der Aalst and Kumar focus on modelling organisational structures and work distribution in the context of teamwork [2]. Concepts from both metamodels, a.o., are considered for the development of an organisational metamodel that could give support to RALTeam [10], a textual notation for the definition of team assignments and team composition compliance rules in business processes. RALTeam takes into account skills and geographical positions of people. Furthermore, it supports all the creation patterns for the description of the team members and some

Task	Evt. class	Task	Evt. class
Register Patient	RP	*Take Blood Sample*	TB
Perform Anamnesis	PA	*Analyse Blood Test*	AB

Table 2: A process alphabet mapped to a log alphabet.

of the advanced resource patterns described by Meyer [29], specifically those related to team selection (Single Entity and Restricted Team Size).

The following conclusions can be drawn from this background: *(i)* to the best of our knowledge, there is no approach available for mining team compositions from process event logs; *(ii)* the process mining approaches addressing the organisational perspective use the workflow resource patterns [33] for the evaluation of their expressive power; *(iii)* the approaches for team assignment and modelling use some of the concepts covered in the workflow resource patterns for the description of teams (e.g., roles and capabilities/skills of the team members) but are also extended towards the direction of social relations and geo-spatial data. We will use this information for the development of our teamwork mining approach.

3 Mining Team Compositions from Event Logs

In this section, we describe our approach to automatically discover team compositions from event logs. Section 3.1 describes the input of our approach and Section 3.2 the produced output. Section 3.3 describes the step of extracting teams and relevant process participants. Section 3.4 specifies the subsequent step of discovering team characteristics.

3.1 Input 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 to organisational units of resources, 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 both *(i)* the enacted task, and *(ii)* the operating resource. Both conditions are commonly respected in real-world event logs [1]. There are different ways of modelling collaborative work in business processes. The most common one, for instance advocated by the BPMN specification [32], is by means of sub-processes, namely activities that are meant to be expanded into a structured workflow of more fine-grained tasks. A sub-process is thus part of an overarching process. It is composed of several tasks and hence, may be performed by a set of people that work towards a specific goal. Sub-processes can

t_1							
$e_{1,1}$	$\preceq$	$e_{1,2}$	$\preceq$	$e_{1,3}$	$\preceq$	$e_{1,4}$	
$\lambda(e_{1,1}) = \text{RP}$ $\rho(e_{1,1}) = i_2$		$\lambda(e_{1,2}) = \text{PA}$ $\rho(e_{1,2}) = i_1$		$\lambda(e_{1,3}) = \text{TB}$ $\rho(e_{1,3}) = i_2$		$\lambda(e_{1,4}) = \text{AB}$ $\rho(e_{1,4}) = i_6$	
t_2							
$e_{2,1}$	$\preceq$	$e_{2,2}$	$\preceq$	$e_{2,3}$	$\preceq$	$e_{2,4}$	
$\lambda(e_{2,1}) = \text{RP}$ $\rho(e_{2,1}) = i_3$		$\lambda(e_{2,2}) = \text{PA}$ $\rho(e_{2,2}) = i_1$		$\lambda(e_{2,3}) = \text{TB}$ $\rho(e_{2,3}) = i_3$		$\lambda(e_{2,4}) = \text{AB}$ $\rho(e_{2,4}) = i_5$	
t_3							
$e_{3,1}$	$\preceq$	$e_{3,2}$	$\preceq$	$e_{3,3}$	$\preceq$	$e_{3,4}$	$\preceq$ $e_{3,5}$
$\lambda(e_{3,1}) = \text{RP}$ $\rho(e_{3,1}) = i_4$		$\lambda(e_{3,2}) = \text{PA}$ $\rho(e_{3,2}) = i_1$		$\lambda(e_{3,3}) = \text{TB}$ $\rho(e_{3,3}) = i_2$		$\lambda(e_{3,4}) = \text{AB}$ $\rho(e_{3,4}) = i_6$	$\lambda(e_{3,5}) = \text{AB}$ $\rho(e_{4,4}) = i_7$
t_4							
$e_{4,1}$	$\preceq$	$e_{4,2}$	$\preceq$	$e_{4,3}$	$\preceq$	$e_{4,4}$	$\preceq$ $e_{4,5}$
$\lambda(e_{4,1}) = \text{RP}$ $\rho(e_{4,1}) = i_2$		$\lambda(e_{4,2}) = \text{PA}$ $\rho(e_{4,2}) = i_1$		$\lambda(e_{4,3}) = \text{TB}$ $\rho(e_{4,3}) = i_2$		$\lambda(e_{4,4}) = \text{AB}$ $\rho(e_{4,4}) = i_6$	$\lambda(e_{4,5}) = \text{AB}$ $\rho(e_{4,4}) = i_6$
t_5							
$e_{5,1}$	$\preceq$	$e_{5,2}$	$\preceq$	$e_{5,3}$	$\preceq$	$e_{5,4}$	$\preceq$ $e_{5,5}$
$\lambda(e_{5,1}) = \text{RP}$ $\rho(e_{5,1}) = i_2$		$\lambda(e_{5,2}) = \text{PA}$ $\rho(e_{5,2}) = i_1$		$\lambda(e_{5,3}) = \text{TB}$ $\rho(e_{5,3}) = i_2$		$\lambda(e_{5,4}) = \text{TB}$ $\rho(e_{5,4}) = i_3$	$\lambda(e_{5,5}) = \text{AB}$ $\rho(e_{4,4}) = i_6$

Table 3: An excerpt of a patient treatment process event log

include other sub-processes or individual tasks, which are performed by one single person. Assuming sub-processes composed of only individual tasks, this definition of collaborative work generates the typical event logs that are used by traditional process mining approaches. For instance, the following excerpt of a patient treatment process event log encoded in the XES logging format [42] shows the recorded information of the event reporting on the execution of task *Perform Anamnesis* performed by resource i_1 (related event attributes are highlighted). Collecting the information from all the tasks in a sub-process we can derive teams of resources that worked together to complete a specific larger activity represented by the sub-process.

```
<event>
   <string key="org:resource" value="i1"/>
   <date key="time:timestamp" value="2013-08-06T14:58:00.000+01:00"/>
   <string key="concept:name" value="Perform Anamnesis"/>
   <string key="lifecycle:transition" value="complete"/>
</event>
```

Formally, let $\mathscr{A}$ be the *process alphabet*, namely the finite non-empty set of activities for a given process. Let $\mathscr{L}$ be the *log alphabet*, namely the finite non-empty set of event classes for a given event log. For the sake of simplicity, we assume that elements of $\mathscr{A}$ and $\mathscr{L}$ coincide by means of a bijective mapping

function *map*, hence for every $a \in \mathscr{A}$ and $\hat{a}, \hat{a}' \in \mathscr{L}$, $a \overset{map}{\longmapsto} \hat{a}$, and if $a \overset{map}{\longmapsto} \hat{a}$, $a \overset{map}{\longmapsto} \hat{a}'$, then $\hat{a} = \hat{a}'$. Therefore, we will henceforth interchangeably adopt the terms "task" and "event class". Table 2 shows an example of mapping from the process alphabet to the event log alphabet.
A trace $\sigma = \langle E, \preceq \rangle$ is a tuple that consists of a finite set of events $E = \{e_1, \ldots, e_{|E|}\}$ on top of which the ordering relation $\preceq \subset E \times E$ is defined. A trace thus designates a finite ordered set of events. Let $\mathfrak{R} = \{i_1, \ldots, i_{|\mathfrak{R}|}\}$ be the *resources universe*, i.e., the finite set of resources i (individuals) for a given process. A finite set of resources $T \subseteq \mathfrak{R}$ will henceforth be named as *team*. Its individuals will be referred to as *team members*.

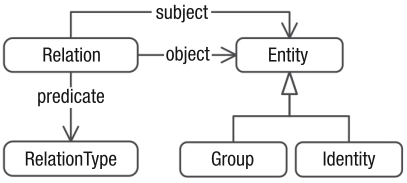
In the light of the above, an event log is formally defined as follows:

Definition 1 (Event log) An event log is a tuple $\Phi = \langle \Sigma, \mathscr{L}, \mathfrak{R}, \lambda, \rho \rangle$ where:

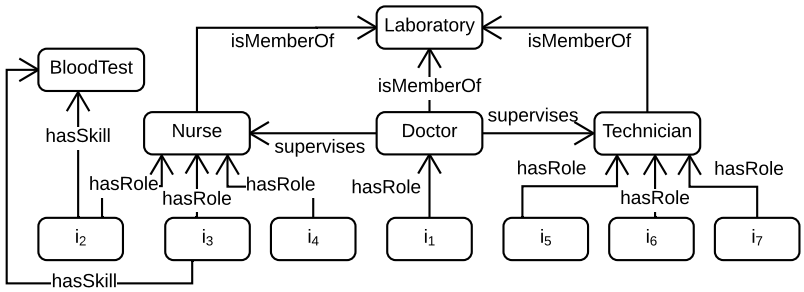
$\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ with $n = |\Sigma|$ is a finite set of *traces*; we denote every trace as $\sigma_i = \langle E_i, \preceq_i \rangle$ with $1 \leqslant i \leqslant n$; the j-th event in trace σ_i is denoted as $e_{i,j}$; we assume that $E_1 \cap \ldots \cap E_n = \emptyset$, and henceforth name $\mathfrak{E} = E_1 \cup \ldots \cup E_n$ as *events universe*;
$\mathscr{L}$ is the *log alphabet* of event classes;
$\mathfrak{R}$ is the *resources universe*;
$\lambda : \mathfrak{E} \to \mathscr{L}$ is a mapping function labelling every event to an event class;
$\rho : \mathfrak{E} \to \mathfrak{R}$ is a mapping function assigning a resource to every event.

Table 3 shows an event log, whose traces consist of events that are labelled by the event classes of Table 2 and assigned to resources $i_1, i_2, \ldots, i_7$. In the remainder of the paper, we will use the shorthand notation $e(a,i)$ to denote an event e such that $\lambda(e) = a$ and $\rho(e) = i$. Table 4 shows the traces of the event log with such a shorthand notation.

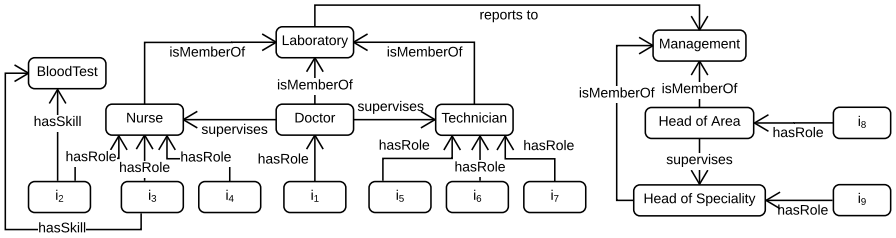
The second input of the technique is an organisational model. Fig. 2a shows the generic organisational meta-model our framework is built on which is based on [9]. Identity represents an individual agent that can be directly assigned to tasks. A Group describes several individuals as a whole. Both Identity and Group are unified under the concept of Entity. A Relation represents the interplay between pairs of Entities, one being a subject, and the other being the object. For Relations, a RelationType specifies its interpretation. The meta-model is well suited for modelling a variety of organisational structures, from common hierarchies to more complex structures like holarchies or federations [22]. Fig. 2b depicts an example of a hierarchical model for a hospital department consisting of five Groups representing the organisational unit (Laboratory), three organisational roles (Nurse, Doctor, Technician) and one skill (BloodTest); and seven instances of Identity (from i_1 to i_7) representing seven individuals. Relations of RelationType *supervises* indicate the hierarchical relations between the roles. Fig. 2c depicts an example of a holarchical model as an extension of the hierarchical model. Holarchies are sets of hierarchies connected with each other [22]. In the figure, the previous hierarchy is linked to the hierarchical structure of another hospital department by a Relation of RelationType *reports to*. A federation is similar to a holarchy but the connection between two hierarchies is centralised on a specific individual, so-called facilitator, mediator



(a) Organisational meta-model



(b) Hierarchical organisational model



(c) Holarchical organisational model

Fig. 2: Organisational meta-model and example organisational models

or broker [22]. A slight variation of the model in Fig. 2c using the Relation of RelationType *reports to* for connecting the Entity *Doctor* and the Entity *Head of Area* instead of *Laboratory* and *Management* would suffice to represent such a structure.

Note that there are several other organisational meta-models that fulfil the necessary requirements to represent complex organisational relations and structures, e.g., [17]. The organisational meta-model is, however, interwoven with the process modelling language we use for representing the mining results (cf. Section 3.2).

Formally, an organisational model is defined as follows.

Definition 2 (Organisational model) An organisational model is a tuple $\Omega = \langle \mathfrak{R}, G, \mathscr{Y}, N \rangle$, where

$\mathfrak{R}$ is the aforementioned resources universe,
G is a set of groups,

$\mathscr{Y}$ is a repertoire of organisational relation types,
$N \subseteq (\mathfrak{R} \cup G) \times \mathscr{Y} \times (\mathfrak{R} \cup G)$ is a relation assigning organisational relation types among individual resources and groups.

In the example above (Fig. 2b):

- $\mathfrak{R} = \{i_1, i_2, \ldots, i_7\}$,
- $G = \{\text{Nurse}, \text{Doctor}, \text{Laboratory}, \text{Technician}, \text{BloodTest}\}$,
- $\mathscr{Y} = \{\text{hasRole}, \text{hasSkill}, \text{memberOf}\}$, and
- N consists of the following tuples:

  $\{i_1, \text{hasRole}, \text{Doctor}\}$, $\{i_2, \text{hasRole}, \text{Nurse}\}$, $\{i_3, \text{hasRole}, \text{Nurse}\}$,
  $\{i_4, \text{hasRole}, \text{Nurse}\}$, $\{i_2, \text{hasSkill}, \text{BloodTest}\}$, $\{i_3, \text{hasSkill}, \text{BloodTest}\}$,
  $\{i_5, \text{hasRole}, \text{Technician}\}$, $\{i_6, \text{hasRole}, \text{Technician}\}$,
  $\{i_7, \text{hasRole}, \text{Technician}\}$, $\{\text{Doctor}, \text{memberOf}, \text{Laboratory}\}$,
  $\{\text{Nurse}, \text{memberOf}, \text{Laboratory}\}$, $\{\text{Technician}, \text{memberOf}, \text{Laboratory}\}$,
  $\{\text{Doctor}, \text{supervises}, \text{Nurse}\}$, $\{\text{Doctor}, \text{supervises}, \text{Technician}\}$.

The two elements of input represent two different perspectives on the process: the event log reports on the registered behaviour of the process instances, and is activity-centric; the process model defines the capabilities and relations of resources and groups of resources, disregarding the behavioural perspective. In the following, we will show how the two perspectives can be integrated in the context of mining.

3.2 Representing Mining Output in the DPIL Mining Framework

For preparing and representing the mining output, we rely on the DPIL mining framework [35]. This framework supports the mining of the organisational perspective of a process, specifically the assignment of resources to process activities and the relation between resources and the control flow of a process. In this way, it can, for instance, be investigated whether certain paths are only allowed for specific types of resources. Note that the presented approach is also applicable with rule-based target languages that support the modelling of organisational patterns with the same or even more expressiveness as DPIL. An application of the proposed approach based on procedural process modelling languages like YAWL [41] is in principle possible, however, would require the development of a corresponding mining approach for the YAWL language first. We will rely on the DPIL framework to develop our teamwork mining approach because it supports several advanced concepts that are useful for mining teams and which are not yet covered by other approaches (see Table 1 above).

The representation of the discovered process models in the framework builds on the Declarative Process Intermediate Language (DPIL) [45], which supports multiple perspectives, among them the resource and the control flow perspectives. 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. In contrast to concrete rules, a rule

```
use group Nurse
use group Doctor
use group Technician
use group Laboratory
use group BloodTest

use relationtype hasSkill
use relationtype hasRole
use relationtype memberOf

process Treatment {
   task Register Patient
   task Perform Anamnesis
   task Take Blood Sample
   task Analyse Blood Test

   ensure role(Perform Anamnesis, Doctor)
   ensure group(Analyse Blood Test, Technician)
   ensure capability(Take Blood Sample, hasSkill, BloodTest)
}
```

Fig. 3: Process with resource assignment rules modelled with DPIL

template consists of placeholders, i.e., typed variables. A rule template is instantiated by providing concrete values for these placeholders.

Templates have formal semantics specified through logical formulae and are equipped either with user-friendly graphical representations or with *macros*. DPIL provides a textual notation based on the use of macros to define reusable rules. For instance, the model representing an exemplary patient treatment described in Fig. 3 makes use of three rule templates represented by the macros *role(A,G)*, *group(A,G)* and *capability(A,RT,G)*. These templates comprise placeholders of type *Task A*, *Group G* and *RelationType RT*. In particular, the model in the figure specifies that the anamnesis must be performed by a resource with the role *Doctor*. Blood sampling must be performed by a resource that has the skills to perform *blood tests*. The blood test must be analysed by a member of the group *Technician*. A set of rule templates to cover, among others, the creation patterns [33], were defined in [35]. The templates that are relevant to our team mining approach are the following:

- The *direct distribution* pattern can be extracted with a *direct(A,I)* template.

  ```
  direct(A,I) iff event(of A) implies event(of A by I)
  ```

- The *role-based distribution* pattern can be extracted with a *role(A,G)* template.

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

- The *capability-based distribution* pattern can be extracted with a *capability(A,RT,G)* template. A capability is represented by a relation of an individual to a group, e.g., i_2 *hasSkill BloodTest*.

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

– The *organisation-based distribution* pattern gives rise to several rule templates. The assignment of resources based on organisational positions or organisational units of individuals can be extracted with an *orgDistSingle(A,RT,G)* template.

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

More specifically, the *group(A,G)* template extracts if a certain task A is performed by a person of a specific organisational unit G.

```
group(A, G) iff
event(of A by :p) implies relation(subject p predicate memberOf object G)
```

By instantiating these rule templates with all possible parameter combinations of defined resources, groups and relation types for the placeholders, it is possible to generate rule candidates that focus on the organisational perspective of the process to be analysed. These candidates can then be verified under consideration of the corresponding organisational model.

3.3 Extraction of Discriminative Teams

The first step is to extract the different sets of participants that are needed to perform the different process instances. Here, the provided event log is scanned once as visualised in Fig. 4. For every trace $\sigma = \langle E, \preceq \rangle$, we extract $T^{\sigma} = \{i \in \mathfrak{R} : e \in E \wedge \rho(e) = i\}$, namely the set of distinct individuals that are associated with at least one of its events. The result of the scan of an event log with n different traces is a set of distinct teams $\Theta = \{T_1,T_2,...,T_m\} \subseteq 2^{\mathfrak{R}}$, with $m \leqslant n$.

These concepts are illustrated in the event log of Tab. 4 which shows five traces of the patient treatment process from above. The provided event log notation denotes the recorded events of a specific task A performed by an individual i with $e(A,i)$ (cf. Section 3.1). According to this notation, $e(\mathrm{RP},i_2)$ specifies an event of activity "Register Patient" (abbreviated as "RP") that has been performed by resource i_2. We show the events in their sequential order here, omitting their real timestamps. For example, the team that performed process instance C1 consists of the distinct set of individuals $\{i_1,i_2,i_6\}$. Since the team of traces C1 and C5 are the same, the scan of the log results in $\Theta = \{\{i_1,i_2,i_6\},\{i_1,i_3,i_5\},\{i_1,i_2,i_4,i_6,i_7\},\{i_1,i_2,i_3,i_6\}\}$.

In real-life event logs some traces may be performed by teams that only occur exceptionally. In order to abstract from rarely occurring teams, we want to extract a set of recurring teams. This problem is similar to finding frequent itemsets based on their support [3,4]. The *support* of an itemset X is the percentage of traces that contain the items of X. Specifically, given a log $\Phi = \langle \Sigma, \mathscr{L}, \mathfrak{R}, \lambda, \rho \rangle$, let $|\Sigma|$ be the total number of traces recorded in the log.

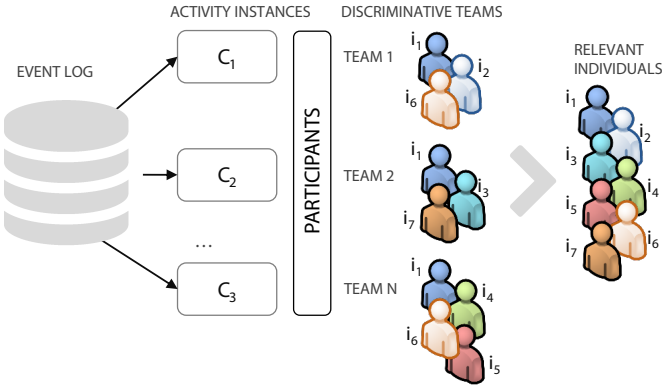


Fig. 4: Extraction of discriminative teams and relevant individuals

Let $\Sigma|_T = \{\sigma \in \Sigma : T^\sigma = T\}$ be the set of traces in Σ performed by a team T. The support value of a team T in log Φ is defined as

$$supp(T) = \frac{|\Sigma|_T|}{|\Sigma|} \tag{1}$$

A team is considered to be relevant if it performs a sufficient number of activities, as reported in logs' traces. Such assessment is indicated by its *support* value being greater than a given threshold *minSupp*. A *minSupp* of 5%, e.g., implies that only teams that occur in at least in 5% of the recorded traces are considered.

3.4 Mining Team Characteristics

The set of frequent teams already provides valuable insights into the operation of a business process. However, it does not help managers to learn how they can compose teams in a potentially more effective way. Therefore, the next step of our approach focuses on the mining of frequent team characteristics. To this extent, we make use of the DPIL Mining Framework and its concepts of rule templates with certain placeholders. These rule templates are used for

ID	Trace	Team
C1	{e(RP,i_2), e(PA,i_1), e(TB,i_2), e(AB,i_6)}	{i_1,i_2,i_6}
C2	{e(RP,i_3), e(PA,i_1), e(TB,i_3), e(AB,i_5)}	{i_1,i_3,i_5}
C3	{e(RP,i_4), e(PA,i_1), e(TB,i_2), e(AB,i_6), e(AB,i_7)}	{i_1,i_2,i_4,i_6,i_7}
C4	{e(RP,i_2), e(PA,i_1), e(TB,i_2), e(AB,i_6)}	{i_1,i_2,i_6}
C5	{e(RP,i_2), e(PA,i_1), e(TB,i_2), e(TB,i_3), e(AB,i_6)}	{i_1,i_2,i_3,i_6}

Table 4: The teams extracted from the traces of an event log

querying the provided event log. A solution to the query is any combination of concrete values for the placeholders that make a concrete rule hold true.

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. All the resulting rule candidates are subsequently evaluated w.r.t. event log $\Phi = \langle \Sigma, \mathscr{L}, \mathfrak{R}, \lambda, \rho \rangle$ (see definition in Section 3.1) to determine the traces where they hold. Consider the *direct*(A,I) template, which specifies that a certain task A is always performed by an individual resource I. Assuming that n_A different tasks and n_I distinct resources occur in the event log (having $0 \leqslant n_A \leqslant |\mathscr{L}|$ and $0 \leqslant n_I \leqslant |\mathfrak{R}|$), there are $n_A \cdot n_I$ rule candidates to be checked, hence $O(|\mathfrak{E}| \cdot |\mathscr{L}| \cdot |\mathfrak{R}|)$ checks to be performed.

In contrast with our previous approach, which mines for resource assignment rules for each single task [35], here we abstract from the tasks and search for assignment rules that refer to a complete sub-process. Hence, the parameter referring to a task in the rule templates described in Section 3.2 can be omitted. This way, the *direct* template, e.g., results in the following definition:

```
direct(I) iff event() implies event(by I)
```

Furthermore, the *role* template then looks as follows:

```
role(G) iff event(by :p) implies relation(subject p predicate hasRole object G)
```

This means, e.g., a *direct*(i_1) rule holds true for a certain process if individual i_1 performs at least one task in every instance of the collaborative activity. Therefore, mining teamwork aspects reduces the number of candidates for the *direct* template to n_I rules to be evaluated, in $O(|\mathfrak{E}| \cdot |\mathfrak{R}|)$ checks. Consider again the example event log in Tab. 5 and the exemplary rule candidates *direct*(i_6), *direct*(i_7) and *group(Technician)*. Checking the *direct* candidates in all the recorded traces reveals that person i_6 only participates in 4 process instances out of 5, and person i_7 only in 1 process instance out of 5. When checking the *group(Technician)* rule, the background knowledge in form of the organisational model needs to be examined. Considering that i_5, i_6, and i_7 are members of the organisational group of *Technician*s, this rule candidate is satisfied in every trace of the example event log.

Evaluating rule candidates as described above yields for every candidate the number of traces in the log where it holds. In order to judge the relevance of the rules, we adopt similar to [28] the support threshold concept from association rule mining for evaluating the relevance of rule candidates. Let $|\Sigma|$ be the

Trace	**direct**(i_6)	**direct**(i_7)	**group**(*Tec.*)
{e(RP,i_2), e(PA,i_1), e(TB,i_2), e(AB,i_6)}	✓	-	✓
{e(RP,i_3), e(PA,i_1), e(TB,i_3), e(AB,i_5)}	-	-	✓
{e(RP,i_4), e(PA,i_1), e(TB,i_2), e(AB,i_6), e(AB,i_7)}	✓	✓	✓
{e(RP,i_2), e(PA,i_1), e(TB,i_2), e(AB,i_6)}	✓	-	✓
{e(RP,i_2), e(PA,i_1), e(TB,i_2), e(TB,i_3), e(AB,i_6)}	✓	-	✓

Table 5: Example event log and satisfaction of exemplary rule candidates

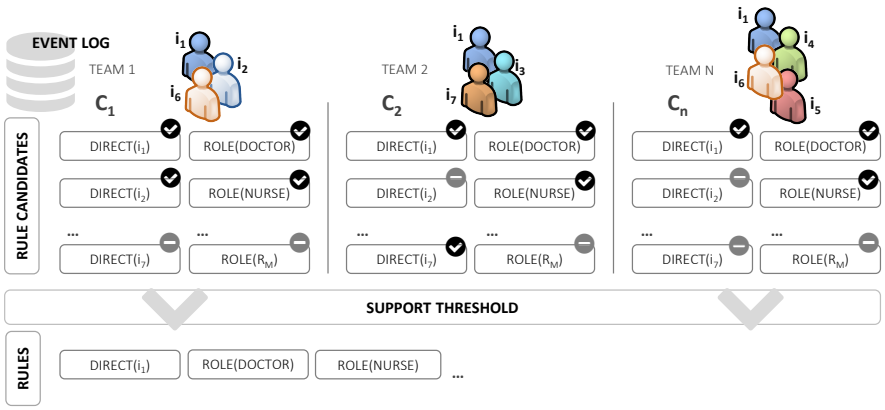


Fig. 5: Checking of various organisational patterns

number of traces in an event log Φ. Let $\Sigma|_r$ be the set of traces in which a rule r is satisfied, and $|\Sigma|_r|$ be its size. The support $supp(r)$ value of rule r is defined as:

$$supp(r) = \frac{|\Sigma|_r|}{|\Sigma|} \tag{2}$$

Based on the support value, we classify a rule candidate r as a *valid* rule ($supp(r) > minSupp_r$) or a *non-valid* rule ($supp(r) < minSupp_r$) with the threshold $minSupp_r$. Using the support values of rule candidates, we can directly extract relevant team characteristics. Considering again the *direct*(i_1) rule from above, its support evaluates to $supp(r) = 1$ while for the *direct*(i_6) rule only a support of $supp(r) = 0.8$ applies. Fig. 5 illustrates the complete mining procedure by example of *direct* and *role* rule candidates.

3.5 Postprocessing

Extracting the characteristics of the team members is not sufficient to describe how the team is actually composed. In order to compose a new team it is necessary to know how the characteristics are distributed among the team members. For instance, the results in Fig. 5 only say that at least *someone* in every team has the extracted characteristics. However, it is not clear how many persons in a team have a certain characteristic and which of these characteristics are maybe fulfilled by one and the same person, i.e., *overlapping* characteristics. In the following, we describe two post-processing steps to define more precisely the way in which a team is composed for a collaborative activity. Each of these steps introduces precision but also computational complexity.

Algorithm 1: Calculating the Minimum Number of Persons

Input: R: Set of discovered rules $R = \{r_1, \ldots, r_{|R|}\}$
Input: Θ: Set of teams $\Theta = \{T_1, \ldots, T_{|\Theta|}\}$
Output: Minimum number of persons min_r that fulfil the characteristics of rule r

```
1  foreach r ∈ R do
2      min_r ← ⊥
3  foreach r ∈ R do
4      foreach T ∈ Θ do
5          currentCount ← 0
6          foreach i ∈ T do
7              if i satisfies characteristic of r then
8                  currentCount ← currentCount + 1
9          if (min_r = ⊥ ∨ currentCount < min_r) ∧ currentCount > 0 then
10             min_r ← currentCount
```

3.5.1 Calculating the Minimum Number of Persons

In a first step we move from analysing at team level to analysing at resource level. Specifically, we count for each characteristic among those extracted for the teams the number of team members that fulfil it. Algorithm 1 describes the procedure. The algorithm takes as an input the set of discovered rules R and the set of extracted teams Θ. The outcome is the minimum number of persons that fulfil a each characteristic, i.e., a rule within a team. At first, the minimum number of persons min_r for every rule r is initialised. The minimum number of persons min_r is calculated for each extracted rule $r \in R$ (line 3). For that, we calculate for every team $T \in \Theta$ (line 4) the number of individuals *currentCount* of $i \in T$ that satisfy the characteristic that is imposed by r (line 6-8). If *currentCount* of the currently examined team is lower than min_r of the current rule r, *currentCount* is assigned to min_r (line 9-10).

3.5.2 Overlapping Characteristics

The previous result is more informative than the initial result because it adds cardinality to the extracted characteristics. However, it is done only at an individual level, i.e., for each single characteristic. Since one single person may have several of the characteristics discovered, in a last step we consider all the possible combinations of characteristics and we check each one of them for each member of the team. Algorithm 2 describes the procedure.

The algorithm takes as an input the set of discovered rules R and the set of extracted teams Θ. The outcome is the set of overlapping rule sets $\overline{R}^o$ with the number of resources that must have each set of characteristics within the team. This offers a more detailed view of the team composition. The algorithm starts with the initialisation of $\overline{R}^o$ (line1). The possible cardinalities for combined rule sets are $\{1, \ldots, |R|\}$. For each cardinality we generate the set of possible combinations of characteristics $\overline{Comb}$ (line 3). In the next step, we analyse

Algorithm 2: Overlapping Characteristics

Input: R: Set of discovered rules $R = \{r_1, \ldots, r_{|R|}\}$
Input: Θ: Set of teams $\Theta = \{T_1, \ldots, T_{|\Theta|}\}$
Output: $\overline{R}^o$: Set of overlapping rule sets $\overline{R}^o = \left\{R^o_1, \ldots, R^o_{|R^o|}\right\}$

```
1  R̄^o ← ∅
2  for j ← 1 to |R| do
3      Comb̄: Set of possible combinations of cardinality j of rules in R
4      foreach C ∈ Comb̄ do
5          R^o ← ∅
6          currentCount ← ⊥
7          foreach T ∈ Θ do
8              currentCountTeam ← 0
9              foreach i ∈ T do
10                 violated ← FALSE
11                 foreach r ∈ C do
12                     R^o ← R^o ∪ {r}
13                     if ¬(i satisfies characteristics of r) then
14                         violated ← TRUE
15                 if ¬(violated) then
16                     currentCountTeam ← currentCountTeam + 1
17             if currentCount = ⊥ ∨ currentCountTeam < currentCount then
18                 currentCount ← currentCountTeam
19         if (currentCount > 0) then
20             R̄^o ← R̄^o ∪ {R^o}
```

each combination $C \in \overline{Comb}$ (line 4). We initialise a variable R^o for the set of overlapping rules and a variable *currentCount* for the minimal number of persons in a team that fulfil all the characteristics in R^o (lines 5-6). Next, we check for each team $T \in \Theta$ the number of persons *currentCountTeam* that fulfil the characteristics in C. Hence, we run through all individuals $i \in T$ (line 9) and check whether a person violates a rule $r \in C$ (lines 11-14). In case that a person i satisfies all characteristics $r \in C$, *currentCountTeam* is incremented (line 15-16). In case we found a new minimal value for the number of persons in a team that fulfil all rules in R^o, *currentCountTeam* is assigned to *currentCount* (line 17-18). After analysing all teams we check if the variable *currentCount* is higher than 0, i.e., if we found a team were there is at least one person that fulfils all characteristics in C (line 19). In that case the set of overlapping rules R^o is added to the result set $\overline{R}^o$ (line 20).

We shortly want to conclude the presented approach. The technique to discover team characteristics and team composition pattern builds upon a declarative process mining approach focusing on the resource perspective that has been extended towards the integration of collaborative activities (Section 3.2). The described two-phase approach first extracts the different teams participating in a collaborative activity (Section 3.3) and then discovers the overall characteristics of the team members in terms of skills, organisational roles,

etc., that are present in these teams (Section 3.4). In Sections 3.5.1 and 3.5.2, we introduced a two-step post-processing phase that derives the most informative team compositions including the discovered characteristics among the team members. In the next section we will describe the implementation and evaluate our approach w.r.t. different real-life event logs.

4 Evaluation

In this section we evaluate the approach in three phases: *(i) implementation*, i.e., we describe if and how it is possible to implement the approach presented so far; *(ii) application*, i.e., we report on the results gathered by applying the approach on several real-life event logs; and *(iii) performance*, i.e., we show the computation time required by the mining procedure.

4.1 Implementation

In this section we describe the implementation of the teamwork mining approach. In order to extract team compositions a potentially big number of rule candidates is likely to be checked w.r.t. the provided event log. The problem of checking a large set of rule candidates can be solved by efficient pattern matching methods like the *rete algorithm* [19]. 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[1] provides an up-to-date 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 and is therefore equivalent to DPIL. The transformation of the most important expressions from DPIL to DRL is shown in Table 6. DPIL rules are translated into DRL rules as shown in row 4. 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 logical implications directly, DPIL implications must be translated into DRL according to the logical equivalence $A \rightarrow B \equiv \neg(A \wedge \neg B)$ (cf. row 5 in the table).

Considering the described transformation rules, the DPIL rule

```
event(by :p) implies relation(subject p predicate hasRole object r)
```

results in the following DRL representation:

[1] Documentations about JBoss Drools is available at `http://docs.jboss.org/drools`

Nr	DPIL expression	DRL expression
1	Identity :i	$i: Identity()
2	Identity I :i	$i: Identity(id == "I")
3	Event(**by** I)	$i: Identity(id == "I") **and** Event(Performer == $i)
4	expr	**rule** Id **when** expr **then** listener.onRuleOccurred(drools.getRule()));
5	x **implies** y	**not** (x **and not** y)

Table 6: Rules for transforming DPIL to DRL expressions

```
rule role(r)
when
 rt: RelationType(Id == "hasRole") and
 g : Group(id == "r") and
  Event(p : Performer) and
  not (Relation(Subject == p, Predicate == rt, Object == g)))
then
 listener.onRuleOccured(drools.getRule());
end
```

The described approach has been implemented as a teamwork mining module in the *DpilMiner* application[2]. Fig. 6 shows the user interface and the different features of the application. The tool guides the user through the teamwork mining procedure in the following way: (i) extracting the different occurring teams w.r.t. to a user-specified *minSupp* threshold as well as calculating some statistical values, such as average team size; (ii) mining team characteristics w.r.t. a user-specified $minSupp_r$ threshold. The characteristics discovered are described as DPIL macros; (iii) providing more informative results about the team compositions by discovering overlaps.

The first step to teamwork mining therefore consists of extracting the set of discriminative teams located on the left hand side of the application. Fig. 6 shows a set of nine different teams extracted from the provided example event log. The user-interface also shows different characteristics of the extracted teams, i.e., in Fig. 6 the minimum team size is three, the maximum team size is four and the average size of all teams is 3.44. After extracting basic team characteristics the user can start team composition mining by clicking the button below. The grid in the middle of the interface shows the extracted team composition rules. In Fig. 6 there are five composition rules and the corresponding support value. The post-processing methods can be initiated through the buttons on the right hand side of the application, i.e., counting the number of people that fulfil a certain team composition characteristic as well as extracting overlapping characteristics. The latter is visualized in the listbox below. The listbox provides the sets of composition rules that are fulfilled by at least one and the same person as well as the number of resources.

[2] A screencast of the DpilMiner is accessible online at `http://miner.kppq.de`.

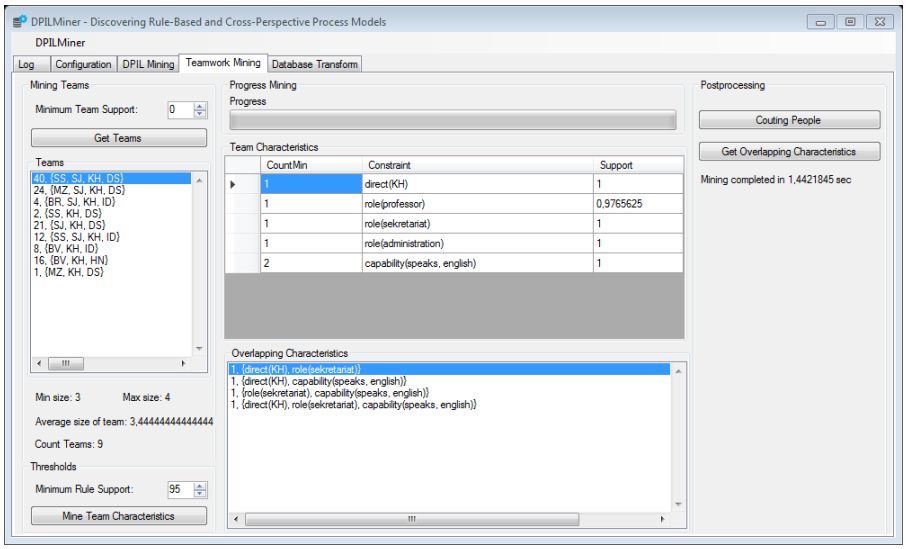


Fig. 6: Teamwork mining module of the DpilMiner

4.2 Application to Real-Life Event Logs

In this section we show that the application of our teamwork mining approach reveals new and previously unknown insights into the composition and the performance of teams of different resources. Therefore, the approach has been applied to the event logs of two real-life processes. First, we describe the use with a university business trip event log where, in addition to the event log, the underlying organisational model is available. Second, we describe the use with administration event logs of five Dutch municipalities where no organisational model is available. In the following, we present our findings.

4.2.1 University Business Trip Management Process

To analyse the complete functionality of our approach we first analysed an event log of a university business trip management system[3]. The log contains 2104 events of ten different tasks related to a process dealing with the application and approval of university business trips as well as the management of accommodations and transfers, e.g., booking hotels, flights or trains. The system has been used for six months by ten employees. The respective organisational model assigns the ten persons to three distinct roles, specifically five *PhD students*, two *professors* and one *secretary*. Furthermore, three employees are members of the group *Administration*. Altogether there are 128 business trip cases, i.e., 128 traces recorded in the log.

[3] The event log is available for download at `http://workbench.kppq.de`

Using the teamwork module of the DpilMiner, we first extracted the different teams w.r.t. several *minSupp* thresholds and did some statistical analyses. Table 7 collects the results of the first mining step. The first column indicates the number of times that a team occurred. The rest of columns show the set of different teams with different *minSupp* values. As they are repeated a great number of times in all three columns, the four first teams (i.e., four first rows) are frequently occurring teams. In particular, they occurred in more than 10% of the recorded cases ($supp > 0.1$). Besides them, some other teams (in the last four rows) have also been recorded. They performed only a small number of cases and hence, they could be considered exceptional teams. Furthermore, taking into account all the occurring teams of the log ($minSupp = 0$), the average size of the teams over those six months was 3.44 members, being the maximum team size of 4 members.

We then executed the second mining step to analyse the extracted teams against the background knowledge provided by the organisational model, and started to apply some post-processing to get more details. Table 8 shows the results of the second mining step and the first post-processing step, i.e., a first approach towards real team compositions in which cardinality is added to each characteristic discovered indicating the minimum number of team members that must have that characteristic. Within each column of the table, the first value indicates such cardinality, the second value shows the characteristic using a DPIL macro and the third value is the $Supp_r$. The first column contains the results with $minSupp_r > 0.95$, which means that a characteristic (i.e., a pattern) is only in the result set if it holds in at least 95% of the recorded traces. The discovered set of patterns reveals that every team mandatorily contains the concrete person *KH*, at least one person with the role *Secretary*, at least one person of the group *Administration* and, in almost every case, at least one person with the role *Professor*. In some exceptional cases, however, the trip management was performed without the participation of a professor ($Supp_r = 0.97$). Furthermore, there are at least two English speakers in every team. The second column shows the results with $Supp_r > 0.6$, i.e., with characteristics

Count	$minSupp = 0$	$minSupp = 0.05$	$minSupp = 0.1$
40	{SS, SJ, KH, DS}	{SS, SJ, KH, DS}	{SS, SJ, KH, DS}
24	{MZ, SJ, KH, DS}	{MZ, SJ, KH, DS}	{MZ, SJ, KH, DS}
21	{SJ, KH, DS}	{SJ, KH, DS}	{SJ, KH, DS}
16	{BV, KH, HN}	{BV, KH, HN}	{BV, KH, HN}
12	{SS, SJ, KH, ID}	{SS, SJ, KH, ID}	
8	{BV, KH, ID}	{BV, KH, ID}	
4	{BR, SJ, KH, ID}		
2	{SS, KH, DS}		
1	{MZ, KH, DS}		
Average Team size	3.44	3.5	3.5
Maximum Team size	4	4	4

Table 7: Extracted teams with different *minSupp* thresholds

$minSupp_r = 0.95$	$minSupp_r = 0.60$
$Count = 1$, direct(KH), $Supp_r = 1$	$Count = 1$, direct(KH), $Supp_r = 1$
	1, direct(SJ), 0.78
	1, direct(DS), 0.68
1, role(Secretary), 1	1, role(Secretary), 1
1, role(Professor), 0.97	1, role(Professor), 0.97
	1, role(Student), 0.64
1, group(Admin), 1	1, group(Admin),1
2, capability(speaks, English),1	2, capability(speaks, English), 1
	1, capability(hasDegree, Master), 0.64

Table 8: Extracted team characteristics

required in a lower number of cases (60%). This relaxation results in a larger number of characteristics found for the teams. In particular, in addition to person *KH*, persons *SJ* and *DS* are usually participating in the process as well, specifically, in 68% and 78% of the traces, respectively. Moreover, in 64% of the traces the team contained at least one person with role *Student* as well as at least one person who has a Master degree. The characteristics that the different teams have in common are shown in the same row in Table 8.

Some team members could actually have more than one of the characteristics discovered. In order to identify such cases, we applied the second post-processing step to detect *overlapping* patterns, as described in Section 3.5.2. The result is shown at the bottom part of the DpilMiner screenshot in Fig. 6. The overlap with the greatest cardinality is found in the fourth line ({direct(KH), role(Secretary), capability(speaks, english)}), which indicates that in each recorded trace one person combines three different discovered characteristics: (i) she is the concrete person *KH*; (ii) she is in role *Secretary* and (iii) she can speak English. The overlaps in the other three lines are contained in that one and therefore do not provide further insights.

4.2.2 Building Permit Process in Municipalities

In this section, we apply our approach to the event logs of a administrative process in five Dutch municipalities. The different event log files[4] contain all building permit applications over a period of approximately four years. The processes in the five municipalities are almost identical. Resources are recorded in the event logs in three different ways: (i) each trace contains an attribute *ResponsibleActor* that captures the person responsible for a certain case; (ii) each event contains an attribute *org:resource* that represents the actual performer of a certain task of a case; and (iii) each event contains an attribute *monitoringResource* that captures resources that were additionally needed to monitor the execution of a certain task. The event log *MunA* contains 1199 traces and 29 distinct resources, *MunB* 1156 traces and 23 resources, *MunC* 1053 traces and 14 resources, *MunD* 1409 traces and 22 resources and *MunE*

[4] DOI: 10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1

	minSupp	0	0.01	0.02	0.03	0.04	0.05	0.1
MunA (29)	Count Teams	331	20	9	5	1	1	0 (max. 0.06)
	Average Team Size	3.77	3.35	3.11	3.4	4	4	-
	Max. Team Size	8	5	4	4	4	4	-
MunB (23)	Count Teams	243	25	9	5	2	1	0 (max. 0.06)
	Average Team Size	3.24	2.88	2.66	2.6	3	3	-
	Max. Team Size	6	4	3	3	3	3	-
MunC (14)	Count Teams	86	17	9	7	4	4	2 (max. 0.27)
	Average Team Size	3.29	3	2.66	2.85	2.75	2.75	3
	Max. Team Size	5	4	4	4	3	3	3
MunD (22)	Count Teams	186	23	16	9	5	3	0 (max. 0.07)
	Average Team Size	3.46	2.91	3.06	3.22	3	3	-
	Max. Team Size	7	4	4	4	4	4	-
MunE (11)	Count Teams	80	25	15	13	9	5	0 (max. 0.09)
	Average Team Size	3.16	2.64	2.6	2.61	2.7	3	-
	Max. Team Size	6	5	4	4	4	4	-

Table 9: Count and average size of discriminative teams

832 traces and 11 resources. We show that teamwork mining discovers interesting results even without the availability of an organisational model.

To process these logs, we combined the different resources that occurred in all the resource attributes and applied the first teamwork mining step over them. Table 9 shows the number of discriminative teams as well as the average and maximum team size for each municipality w.r.t. different *minSupp* thresholds. The table shows several interesting insights that can be described by looking at *MunA*, *MunC* and *MunE*:

(i) To complete the same process, *MunA* needs significantly bigger teams compared to other municipalities. For instance, in *MunA* the maximum team size is 8 without threshold and 4 with increased *minSupp*, whereas teams in *MunC* have a maximum size of 5 and 3, respectively.

(ii) The number of discriminative teams in *MunA* (331) is four times greater than in *MunC* (86) and *MunE* (80).

(iii) Related to the previous point, the team variability is much higher in *MunA* than in *MunC* and *MunE*, i.e., teams tended not to be repeated over time. Specifically, in *MunA* only 1 team occurred in more than 4% of recorded cases ($Supp > 0.04$), in contrast to 9 teams in *MunE*. This is also evidenced by the leap from 331 teams to 20 teams from $minSupp = 0$ to $minSupp > 0.01$ in *MunA*, i.e, most of the teams performed only a very small number of cases. *MunC* represents the other extreme: 2 teams performed more than 10% of the recorded cases and the maximum *support* found for one team was 0.27, which means that a certain team performed almost one third of all cases.

We then analysed the characteristics of the extracted teams. Since there is no background knowledge available in the form of an organisational model, we could only analyse which persons were most frequently required to be part in a team and hence, to participate in the process. The results of the

MunA	MunB	MunC	MunD	MunE
direct(560872),$Supp_r$=0.64	560602, 0.62	560752, 0.87	560696, 0.76	560532, 0.64
direct(560890),$Supp_r$=0.52		560781, 0.78	560673, 0.56	560458, 0.62
				560530, 0.56

Table 10: Required participation of certain resources

analysis are depicted in Table 10 and underline the findings of the previous analysis. Consider the first column with the resulting *direct* rules of *MunA*: even though team variability is high, the resources with ID 560872 and 560890 play an important role in the execution of the process as they are part of the performing teams in 64% and 52% of the executions, respectively. When looking at *MunC*, on the other hand, the results show a strong dependency of process execution w.r.t. the resources with ID 560752 and 560781. Here, these two persons took part in 87% and 78% of cases, respectively. The lower degree of team variability in *MunC* leads to an increasing dependency of certain resources and increases the risk of running into bottlenecks.

In a last step, we analysed the performance, i.e., the average processing time rounded up to full days of whole process instances, in each municipality. Therefore, we compared the five most relevant teams w.r.t. the number of performed cases. The list of teams, the corresponding resources and the average performance is given in Table 11. The table shows that *(i)* the processing performances in the municipalities are significantly different and that *(ii)* different teams in each municipality performed cases faster and slower. When considering for example *MunA*, *MunB* and *MunC* it is significant that teams in *MunC* perform on average much faster than *MunA* and *MunB*. *MunB* depicts the other extrem where all teams have an average processing time of more than one hundred days. Consider furthermore the different teams of *MunC*. Here, the fastest team (line 4) performed cases on average in 19.5 days while the slowest team (line 1) performed cases on average in 61 days, even though the fastest team only consists of two resources compared to four resources of the slowest team.

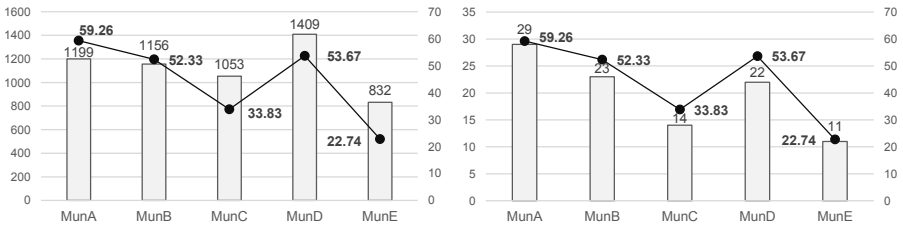
4.2.3 Performance Evaluation

In this section, we check the computational performance of our teamwork mining implementation using the five event logs that have already been used in Section 4.2.2. For each event log we analysed the time to check the mandatory participation of certain resources, i.e., the time to check all possible rule candidates of the *direct* template. All the computation times reported in this section are measured on a Core i7 CPU @2.80 GHz with 8 GB Ram. The results of the performance evaluation are illustrated in Fig. 7. Specifically, Fig. 7a shows the execution time in seconds (right-hand-side axis) w.r.t. the number of traces of the different logs (left-hand-side axis). Fig. 7b shows the execution time w.r.t. the number of distinct resources that appear in each event log. The re-

Log	Team Composition	Num. of instances	Avg. Performance [days]
MunA (29)	560872, 2670601, 560890	45	50.9
	2670601, 560872, 560912	44	134.0
	560872, 560464, 3273854, 560890	83	43.6
	560872, 2670601, 560890, 560464	39	72.0
	560872, 3273854, 560464	41	89.9
MunB (23)	560458, 560532, 560521, 560530	51	147.9
	560458, 560530	59	211.1
	560519, 560532	47	104.5
	560519, 560530, 560532, 560458	69	262.6
	560458, 560532, 560530	50	379.0
MunC (14)	560741, 560696, 560454, 560673	101	61.0
	560673, 560696	82	29.9
	560673, 560696, 560454	112	49.4
	560696, 560749	58	19.5
	560749, 560454, 560696, 560741	67	52.7
MunD (22)	560752, 1550894, 560781	139	41.0
	560781, 560852	42	28.9
	560752, 560852, 560781	285	45.8
	560752, 560852	54	57.9
	560812, 560781, 560752	90	15.3
MunE (11)	560604, 560602	35	74.4
	560604, 8492512, 560602	35	39.4
	1254625, 560604, 560602	91	85.8
	1254625, 560604	35	106.2
	560602, 560600, 1254625	49	80.8

Table 11: Average case runtime for most relevant teams

sults show that the execution time of the approach is primarily determined by the number of distinct resources, i.e., by the number of rule candidates to be checked and not by the amount of traces. This is highlighted by the bars of *MunA*, *MunC* and *MunD*. Even if the number of traces is considerably higher in *MunD* (1409) compared to *MunA* (1199), the time for analysing *MunD* is 5.59 sec lower than for *MunA*. Note that the number of distinct resources in *MunA* (29) is 24% higher than in *MunD* (22). In addition, the time for analysing the 1053 traces of *MunC* (33.83 sec) is also clearly determined by the number of resources (14).



(a) Performance w.r.t. number of traces (b) Performance w.r.t. resources

Fig. 7: Performance analysis of the teamwork mining approach

In short, the mining can be performed in less than 1 minute for all the five real-life event logs, which underlines the practical applicability of our implementation. Note that the extraction of teams and their general characteristics is performed directly when importing the event log. The import time takes in all cases only 1 to 2 seconds. The post-processing is independent from the size of the event logs since it only works on the available resources and the organisational model. Even if the post-processing algorithms have a high complexity (cf. Section 3.5), their execution time is rather low due to the usually small number of resources. For instance, the application on the business trip event log resulted in a post-processing time of less than 1 sec.

5 Conclusions, Limitations and Future Work

In this paper we presented a process mining framework to discover team attributes and composition patterns of collaborative activities in business processes. The approach builds upon a declarative process mining approach focusing on the process resource perspective that has been extended towards the integration of collaborative activities. The described two-phase approach first extracts the different teams participating in a collaborative activity from an event log and then discovers the overall characteristics of the team members in terms of skills, organisational roles, etc., that are present in these teams. A subsequent two-step post-processing phase derives the most informative team compositions including the discovered characteristics among the team members. In our evaluation with real-life event logs from the university and the public administration domains we tested its practical applicability and runtime performance. In both application areas we showed that teamwork mining provides interesting insights in the way teams are composed and how collaborative work is performed. Furthermore, we extracted correlations between different team compositions and the process performance.

We shortly want to reflect on relevant requirements of our technique. The presented teamwork mining approach takes as input a process execution event log and optionally an organisational model, i.e., prior knowledge about the roles, capabilities, and the membership to organisational units of resources. We require that events contain an explicit reference to both the enacted activity, and the operating resource. Both conditions are commonly respected in real-world event logs. Our evaluation underlines that more complex team composition rules like, e.g., necessary resources with certain roles and capabilities, can only be extracted if an organisational model of the considered organisation is available. Basic team characteristics, i.e., number and size of discriminative teams as well as most frequently participating resources, however, can be discovered by only analysing a provided process execution log without organisational background knowledge.

The quality of extracted team composition rules is strongly related to the specificity of facts defined in the organisational model. If the organisational model contains irrelevant relations (e.g., a relation “speaks English”, if every-

one in the organisation speaks English) then our approach will inevitably also discover these spurious rules, i.e., every extracted team will contain an English speaker. This issue can be approached by controlling and optimising the specificity of facts of the organisational relations modelled in the underlying organisational background knowledge.

In future work we want to apply our approach on event logs from additional domains like hospital logs and use the results in different settings, e.g., for checking compliance rules with respect to team compositions.

References

1. van der Aalst, W.: Process mining: discovery, conformance and enhancement of business processes. Springer (2011)
2. van der Aalst, W.M.P., Kumar, A.: A Reference Model for Team-enabled Workflow Management Systems. Data Knowl. Eng. **38(3)**, 335–363 (2001)
3. Adamo, J.: Data mining for association rules and sequential patterns - sequential and parallel algorithms. Springer (2001)
4. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Int. Conference on Management of Data, pp. 207–216 (1993)
5. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Power in Unity: Forming Teams in Large-scale Community Systems. In: CIKM, pp. 599–608 (2010)
6. Baumgrass, A., Strembeck, M.: Bridging the gap between role mining and role engineering via migration guides. Inf. Sec. Techn. Report **17**(4), 148–172 (2013)
7. Baykasoglu, A., Dereli, T., Das, S.: Project Team Selection Using Fuzzy Optimization Approach. Cybern. Syst. **38**(2), 155–185 (2007)
8. Bose, J.C., Maggi, F.M., van der Aalst, W.: Enhancing Declare Maps Based on Event Correlations. In: BPM, pp. 97–112 (2013)
9. Bussler, C.: Organisationsverwaltung in Workflow-Management-Systemen. Dt. Univ.-Verlag (1998)
10. Cabanillas, C., Resinas, M., Mendling, J., Cortés, A.R.: Automated team selection and compliance checking in business processes. In: ICSSP, pp. 42–51 (2015)
11. Datta, A., Yong, J.T.T., Ventresque, A.: T-RecS: team recommendation system through expertise and cohesiveness. In: WWW, pp. 201–204 (2011)
12. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive Processes: Characteristics, requirements and analysis of contemporary approaches. J. Data Semantics **4**(1), 29–57 (2015)
13. Dorn, C., Dustdar, S.: Composing near-optimal expert teams: a trade-off between skills and connectivity. In: OTM, pp. 472–489. Springer (2010)
14. Dorn, C., Skopik, F., Schall, D., Dustdar, S.: Interaction mining and skill-dependent recommendations for multi-objective team composition. Data & Knowledge Engineering **70**(10), 866–891 (2011)
15. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013)
16. Dustdar, S.: Caramba: Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams. Distributed and Parallel Databases **15**(1), 45–66 (2004)
17. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: an organizational view of multi-agent systems. In: Agent-Oriented Software Engineering IV, pp. 214–230. Springer (2003)
18. Fitzpatrick, E.L., Askin, R.G.: Forming Effective Worker Teams with Multi-functional Skill Requirements. Comput. Ind. Eng. **48**, 593–608 (2005)
19. Forgy, C.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial intelligence **19**(1), 17–37 (1982)

20. Guimarães, N., Antunes, P., Pereira, A.: The Integration of Workflow Systems and Collaboration Tools. In: Workflow Management Systems and Interoperability, *NATO ASI Series*, vol. 164, pp. 222–245. Springer (1998)
21. Hanachi, C., Gaaloul, W., Mondi, R.: Performative-Based Mining of Workflow Organizational Structures. In: C. Huemer, P. Lops (eds.) EC-Web, pp. 63–75 (2012)
22. Horling, B., Lesser, V.: A Survey of Multi-agent Organizational Paradigms. Knowl. Eng. Rev. **19**(4), 281–316 (2004)
23. Jin, T., Wang, J., Wen, L.: Organizational modeling from event logs. In: GCC, pp. 670–675 (2007)
24. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The Robot World Cup Initiative. In: International Conference on Autonomous Agents, pp. 340–347 (1997)
25. Lappas, T., Liu, K., Terzi, E.: Finding a Team of Experts in Social Networks. In: KDD, pp. 467–476 (2009)
26. Leitner, M., Baumgrass, A., Schefer-Wenzl, S., Rinderle-Ma, S., Strembeck, M.: A Case Study on the Suitability of Process Mining to Produce Current-State RBAC Models. In: BPM Workshops, pp. 719–724 (2012)
27. Ly, L.T., Rinderle, S., Dadam, P., Reichert, M.: Mining staff assignment rules from event-based data. In: BPM Workshops, pp. 177–190 (2006)
28. Maggi, F.M., Bose, J.C., van der Aalst, W.: Efficient Discovery of Understandable Declarative Process Models from Event Logs. In: CAiSE, pp. 270–285 (2012)
29. Meyer, A.: Resource Perspective in BPMN - Extending BPMN to Support Resource Management and Planning. Master's thesis, Hasso Plattner Institute (2009)
30. Nakatumba, J., van der Aalst, W.: Analyzing resource behavior using process mining. In: BPM Workshops, pp. 69–80 (2010)
31. Nurcan, S.: Analysis and design of co-operative work processes: a framework. Information and Software Technology **40**(3), 143–156 (1998)
32. OMG: BPMN 2.0. Recommendation, OMG (2011)
33. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: CAiSE, pp. 216–232 (2005)
34. Salas, E., Stagl, K.C., Burke, C.S., Goodwin, G.F.: Fostering team effectiveness in organizations: toward an integrative theoretical framework. Nebraska Symposium on Motivation **52**, 185–243 (2007)
35. Schönig, S., Cabanillas, C., Jablonski, S., Mendling, J.: Mining the Organisational Perspective in Agile Business Processes. In: BPMDS, vol. 214, pp. 37–52 (2015)
36. Singh, P.V.: The Small-world Effect: The Influence of Macro-level Properties of Developer Collaboration Networks on Open-source Project Success. ACM Trans. Softw. Eng. Methodol. **20**(2), 6:1–6:27 (2010)
37. Song, M., Van der Aalst, W.M.: Towards comprehensive support for organizational mining. Decision Support Systems **46**(1), 300–317 (2008)
38. Tambe, M.: Teamwork in real-world, dynamic environments. In: International Conference on Multi-Agent Systems (ICMAS-96). AAAI Press (1996)
39. Tambe, M., Adibi, J., Al-Onaizan, Y., Erdem, A., Kaminka, G.A., Marsella, S.C., Muslea, I.: Building agent teams using an explicit teamwork model and learning. Artificial Intelligence **110**(2), 215–239 (1999)
40. Van Der Aalst, W.M., Reijers, H.A., Song, M.: Discovering Social Networks from Event Logs. CSCW **14**(6), 549–593 (2005)
41. Van Der Aalst, W.M., Ter Hofstede, A.H.: Yawl: yet another workflow language. Information systems **30**(4), 245–275 (2005)
42. Verbeek, E., Buijs, J., van Dongen, B., van der Aalst, W.: XES, xESame, and ProM 6. In: Information Systems Evolution, pp. 60–75 (2011)
43. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2012)
44. Yang, D.N., Chen, Y.L., Lee, W.C., Chen, M.S.: On Social-temporal Group Query with Acquaintance Constraint. Proc. VLDB Endow. **4(6)**, 397–408 (2011)
45. Zeising, M., Schönig, S., Jablonski, S.: Towards a Common Platform for the Support of Routine and Agile Business Processes. In: Collaborative Computing: Networking, Applications and Worksharing, pp. 94–103 (2014)
46. Zhao, W., Zhao, X.: Process Mining from the Organizational Perspective. In: Foundations of Intelligent Systems, pp. 701–708 (2014)