TU Eindhoven, Computer Science Report, Beta Working Paper Series, No. 213, 2007

Life-Cycle Support for Staff Assignment Rules in Process-Aware Information Systems*

Stefanie Rinderle-Ma[1,2] and Wil M.P. van der Aalst[2]

[1]Department Databases and Information Systems,
Faculty of Engineering and Computer Sciences,
Ulm University, 89069 Ulm, Germany.
[2]Department of Technology Management,
Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
stefanie.rinderle@uni-ulm.de, w.m.p.v.d.aalst@tm.tue.nl

Abstract. *Process mining* has been proposed as a tool for analyzing business processes based on events logs. Today, most information systems are logging events in some log and thus provide detailed information about the processes they are supporting. This information can be used for two forms of process mining: conformance checking (comparing the actual process with some a-priori model) and discovery (deriving a model from scratch). Most of the process mining tools have been focusing on the control-flow perspective and today it is possible to automatically construct process models that can be used for the configuration of *Process-Aware Information Systems* (PAISs). This paper provides an overview of process mining and focuses on a neglected aspect of PAISs: staff assignment. We propose an approach for *staff assignment mining* based on decision tree learning, i.e., based on some organizational model and an event log we try to discover allocation rules. This is useful for configuring new PAISs. However, it can also be used to evaluate staff assignment rules in some existing PAIS. Based on this, flaws and redundancies within staff assignment rules (e.g., security holes by offering process activities to non-authorized users in exceptional cases) can be detected and optimization strategies can be derived automatically. The approach has been implemented in the context of the ProM framework and different strategies have been evaluated using simulation. Altogether, this work contributes to a complete life-cycle support for staff assignment rules.

1 Introduction

New trends in information technology and developments at the (e-business) market let companies crave for automated business process support. Process-Aware

* We thank the EIT for supporting the research stay of Stefanie Rinderle-Ma at the Department of Technology Management, Eindhoven University of Technology, during which this work was conducted.

Information Systems (PAISs) [25] (e.g., in the form of a workflow management system [4]) allow for the explicit definition of the process logic, the execution and monitoring of processes, the integration of distributed application components, and the provision of worklists to authorized users.

1.1 Problem Description

Despite the promising perspectives of PAISs, their breakthrough with respect to a broad application in practice has not taken place so far. One reason might be that, in general, the discovery and the design of business processes are difficult tasks: It requires deep knowledge of the business process at hand (i.e., lengthy discussions with the workers and management ar needed) and the process modeling language being used. In this context, *process mining* technology has brought up a very promising alternative to classical process modeling approaches. Based on collecting audit trail data of already executed processes it is possible to automatically derive the process control-flow (note that in most cases, prior to the deployment of a PAIS, the processes are already there). Closely monitoring the events taking place at runtime also enables *process diagnosis*, i.e., detecting flaws or discrepancies between the design constructed in the design phase and the actual execution registered in the enactment phase (cf. Figure 1). Coupled with adaptive process management technology [1, 38, 49, 60] a complete support of the business process life-cycle becomes possible.

By developing a variety of algorithms and implementing them within the ProM framework [22], process mining has become a mature technology in the last years, in particular with focus on control-flow mining [11, 6]. However, further aspects have to be specified in order to execute the processes in the sequel. One important task is to model the related organizational structures capturing the agents working in the associated domain, their roles and abilities as well as the organizational units they belong to (e.g., `Agent_1` having role `Role_1` and belonging to organizational unit `OU_1`). Social network mining [8, 7] constitutes a first approach for analyzing audit trail data with respect to agents and their relation between each other (e.g., who hands over work to whom). Although this approach enables us to gain insight into the organizational structures behind the audit trail data, it does not provide information about how process models and organizational models are coupled. In today's PAISs this is accomplished by defining *staff assignment rules* which link the process activities to elements of the organizational structures (cf. Figure 1). At runtime, the system can determine which agents are authorized to work on certain activities by resolving the staff assignment rules over the underlying agent set. Accordingly, the system puts the activities to be processed into work lists of qualified agents. Based on this mechanism the controlled execution of process activities (i.e., by authorized agents only) is ensured.

As discussed the definition of staff assignment rules is indispensable for the execution of business processes. Similar to the discovery and definition of business processes the specification of staff assignment rules might often be a complex task for users. Therefore the automated discovery and design of staff assignment

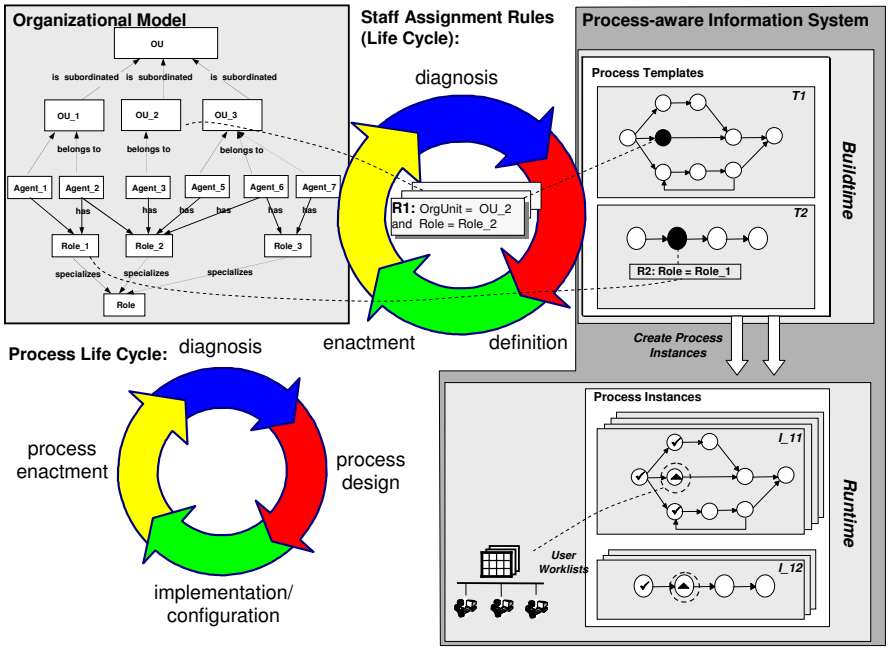


Fig. 1. Life-cycles in process-aware information systems.

rules could be of high interest in practice. In [39] the problem of mining staff assignment rules from audit trail data was addressed for the first time. Based on existing organizational structures and audit trail data, a decision tree based algorithm automatically determines the staff assignment rule for a certain activity.

Analogous to process models, the specification of staff assignment rules may not be stable forever (i.e., deviations from the pre-defined rules may occur rather frequently). Therefore a diagnosis phase is included within the life-cycle of staff assignment rules where discrepancies between pre-defined rules and actual behavior of agents regarding the execution of process activities is analyzed. Consider, for example, Figure 2 where activity x is actually performed by a subset of agents qualifying for the pre-defined staff assignment rule for x.

However, it is not sufficient to just detect such discrepancies. The diagnosis phase should also lead to suggestions for optimizing the staff assignment rules in the sequel: First of all, the pre-defined staff assignment rules may contain *redundant* parts (i.e., the set of agents qualifying for the affected staff assignment rule does not change if the redundant part is removed). However, at runtime these redundant parts are evaluated as well since the system cannot distinguish between necessary and unnecessary parts of staff assignment rules. This, in turn, might lead to performance losses, in particular when resolving staff assignment rules for a large number of running process instances (for large hospitals, for example,

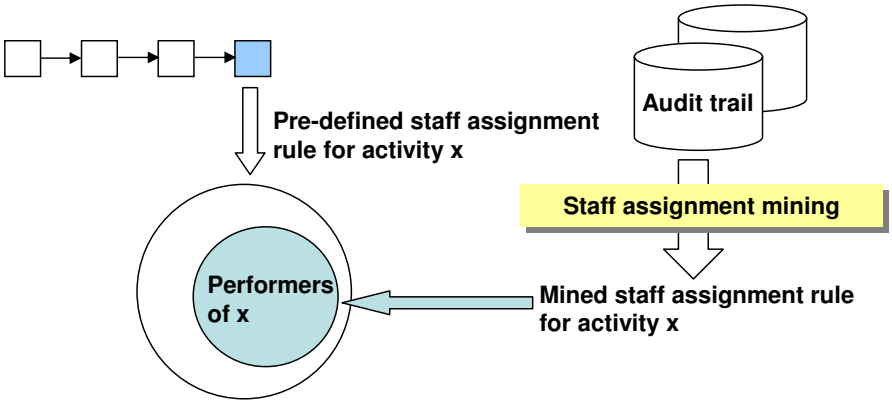


Fig. 2. Deriving and evaluating staff assignment rules

10000 process instances may be active at the same time). A second optimization for staff assignment rules is based on the detection of agents which have executed activities in an exceptional manner (e.g., agents substituting others in case of holidays or diseases). Deriving specifications of substitution assignments (e.g., the examination of doctor Smith is always substituted by doctor Black) would contribute to controlled handling of such exceptional situations (i.e., the examination is intentionally offered to Black if Smith is not available and not wrongly to nurse Johnson). Thirdly, similar to business processes, staff assignment rules may be specified in a sub-optimal way, or change over time. In this case, at least, the system should be able to detect the modeling flaws and report them to users. For all these reasons an adequate support for discovering, designing, and diagnosing staff assignment rules during their life-cycle would be very beneficiary for the practical application of PAISs.

1.2 Contribution

In this paper we provide a framework for the discovery and design as well as for the diagnosis and optimization of staff assignment rules. It is based on a decision tree based approach and a simulation which is evaluated by different plug-ins of the ProM framework[1]. With these results a comprehensive support of the staff assignment life-cycle (cf. Figure 1) becomes possible.

First of all, we conduct a complete simulation for an example medical treatment process using CPN tools and ProMimport [33]. Based on the simulation data, a complete overview of existing process mining techniques is provided ranging from control-flow mining [11], social network mining [8, 7], and decision mining to analysis techniques like conformance checking [50] and property checking [2]. This constitutes a complete overview of existing techniques for process

[1] This analysis constitutes a substantial extension of the work presented in [39] where the basic staff assignment mining concepts have been introduced.

life-cycle support. Further on we provide formal definitions for organizational models and for staff assignment rules as well as the decision tree based mining approach presented in [39].

After introducing the fundamental definitions the simulation data is evaluated with respect to staff assignment mining. For the analysis we assume predefined staff assignment rules and compare them with the mining results. Based on the results, possible relations between original and mined staff assignment rules are formalized. In addition, we show how these relations can be used for deriving optimization suggestions in the sequel. In the following we extend the simulation scenario by introducing exceptional agent behavior (i.e., agent substitutions for a certain percentage of process instances). By evaluating the simulation data we show how substitution rules can be automatically specified. Altogether the presented approach provides complete life-cycle support for staff assignment rules and therefore contributes to the practical applicability of PAISs.

The remainder of this paper is organized as follows: In Section 2 we provide an overview of the existing process mining techniques based on simulation data. Section 3 contains background information on defining organizational structures as well as on the basic staff assignment mining approach. The evaluation framework for staff assignment rules including optimization suggestions is presented in Section 4. We discuss related work in Section 5 and conclude with a summary of the presented results and an outlook on future work in Section 6.

2 Process Mining

This section provides an overview of process mining. Using a running example, it shows the different ways to discover and check the various perspectives of a process based on some event log.

2.1 Overview

Today's information systems are logging events that are stored in so-called "event logs". For example, any user action is logged in ERP systems like SAP R/3, workflow management systems like Staffware, and case handling systems like FLOWer. Classical information systems have some centralized database for logging such events (called transaction log or audit trail). Modern service-oriented architectures record the interactions between web services (e.g., in the form of SOAP messages). Moreover, today's organizations are forced to log events by national or international regulations (cf. the Sarbanes-Oxley (SOX) Act that is forcing organizations to audit their processes). As a result of these developments, there is an abundance of process-related data available. The goal of process mining is to use this data, i.e., enable a fine grained analysis of processes based on event logs.

Figure 3 illustrates the basic idea of process mining. As indicated a wide variety of operational process are supported by information systems that record events. Crucial for process mining is the existence of an event log showing in

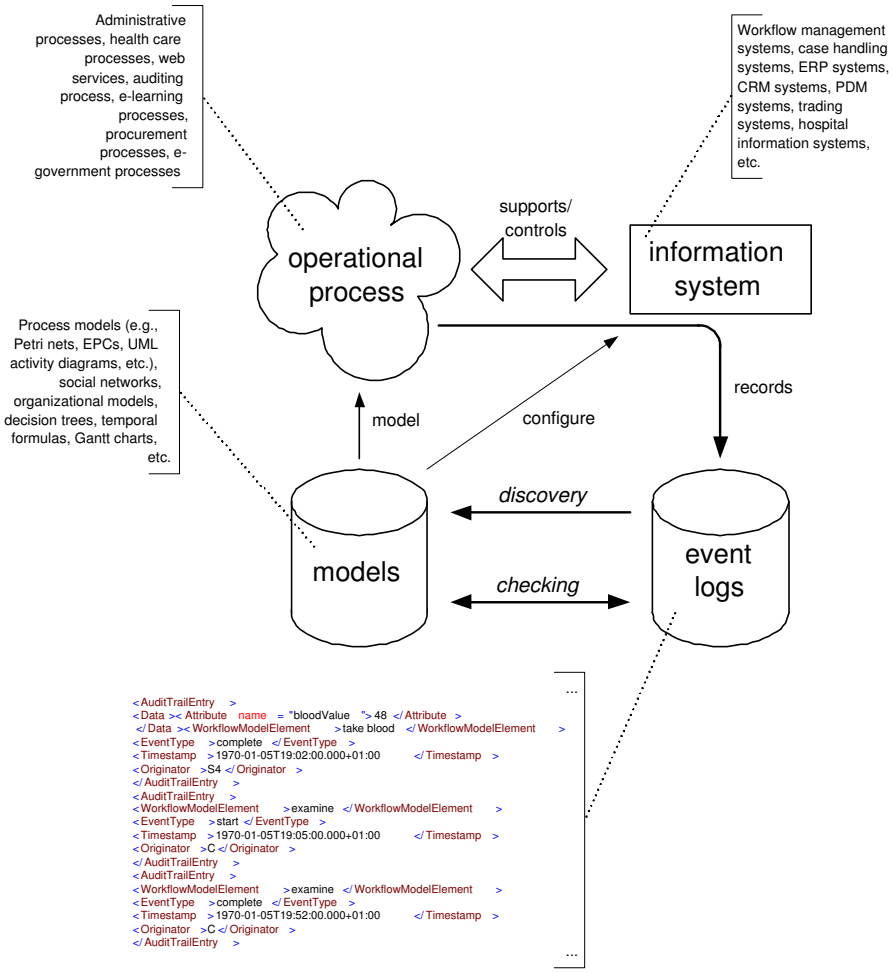


Fig. 3. An overview of process mining.

chronological order relevant events, e.g., events like "payment for customer order XQ665456 is received", "X-ray taken for patient with id 86868", or "building permit BP085354 is granted". These events are recorded in a so-called *event log*. Based on such an event log two types of analysis are possible: *discovery* and *checking*. Discovery algorithms try to extract models from event logs without any a-priori information. For example, the α algorithm [11] is able to discover a Petri net model capturing the control-flow of a process based on an analysis of the log. Other approaches assume some a-priori model and check whether the log and the model fit together. For example, it is possible to formulate some business rule in Linear Temporal Logic (LTL) and then check whether the behavior recorded in the log is consistent with this rule [2]. Orthogonal to the dimension distinguishing discovery (no a-priori model) and checking (some a-priori model is used as a reference) is the dimension "distinguishing the various perspectives of operational processes. In this paper, we distinguish the *control-flow*, *data*, and *organizational perspectives*. The control-flow perspective is mainly concerned with the selection and ordering of activities. Typical languages to describe this perspective are Petri nets, Event-driven Process Chains (EPCs), UML activity diagrams, etc. The data perspective is more concerned with the information related to activities and cases (i.e., process instances). Typical examples of models in this area are decision trees and data models. The organizational perspective focuses on the agents (typically people but in principle also other types of resources) executing the activities. Using these two dimensions, we can identify 2*3=6 process mining classes: discovery/checking and control-flow/data/organization. The α algorithm [11] mentioned earlier is a control-flow discovery algorithm and fits into one of these 6 classes. Although some process mining approaches cover multiple classes, most techniques can be positioned in one of the 6 classes.

The focus of this paper is on staff assignment mining. This fits primarily in the organizational discovery class. However, before focusing on staff assignment mining we provide an overview of process mining using a running example.

2.2 Running Example

To illustrate various classes of process mining, we use the treatment process shown in Figure 4. First of all, the patient is registered (activity `registration`) by his or her name (data element `patientName`) which takes approximately 15 minutes. Afterwards the examination of the patient (activity `examine`) and taking a blood sample (activity `take blood`) can be executed in parallel. Both activities require the patient name as input data. Activity `take blood` writes data element `bloodValue` when completed. The expected duration for activity `examine` is 1 hour whereas taking blood takes approximately 15 minutes. After taking the blood sample and examining the patient an evaluation of the blood value takes place (activity `examine`). If the blood value exceeds 50 units the patient has to undergo a `special therapy` with a duration of 4 hours. Otherwise (if the blood values is equal or less 50 units) the patient participates in a `group therapy`. The expected duration of the group therapy is 2 hours. If the patient has undergone either the special or the group therapy a report is written

(activity `write report`) with an expected duration of 30 minutes. Finally, the rooms for the patient treatment are locked (activity `lock rooms`) which takes approximately 30 minutes.

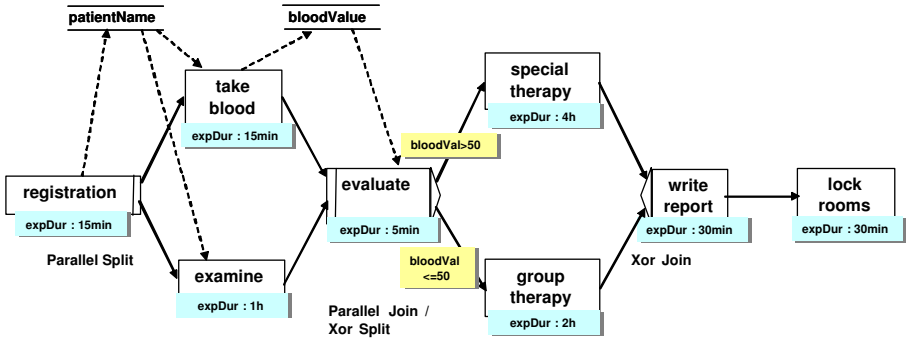


Fig. 4. Running example represented in ADEPT notation (for details on this notation see [46]).

Now we assume that we do not know this process, i.e., we can only see events related to the activities shown in Figure 4. These are the events recorded by the corresponding information system in some *event log*. A fragment of such an event log could be:

```
<WorkflowLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
...
    <Process id="Treatment Process"
      description="log contains information on patients">
    ...
        <ProcessInstance id="567"
          description="Event related to patient John Smith">
            <AuditTrailEntry>
              <WorkflowModelElement>registration</WorkflowModelElement>
              <EventType >offer</EventType>
              <Timestamp>2005-01-01T01:00:00.000+01:00</Timestamp>
              <Originator></Originator>
            </AuditTrailEntry>
            <AuditTrailEntry>
              <WorkflowModelElement>registration</WorkflowModelElement>
              <EventType >start</EventType>
              <Timestamp>2005-01-01T01:00:00.000+01:00</Timestamp>
              <Originator>S7</Originator>
            </AuditTrailEntry>
            <AuditTrailEntry>
              <Data>
```

```
            <Attribute name = "patientName">John Smith </Attribute>
            </Data>
            <WorkflowModelElement>registration</WorkflowModelElement>
            <EventType >complete</EventType>
            <Timestamp>2005-01-01T01:16:00.000+01:00</Timestamp>
            <Originator>S7</Originator>
          </AuditTrailEntry>
          ...
          <AuditTrailEntry>
            <Data>
            <Attribute name = "bloodValue">34 </Attribute>
            </Data>
            <WorkflowModelElement>take blood</WorkflowModelElement>
            <EventType >complete</EventType>
            <Timestamp>2005-01-01T01:31:00.000+01:00</Timestamp>
            <Originator>D7</Originator>
          </AuditTrailEntry>
          ...
      </ProcessInstance>
      ...
  </Process>
</WorkflowLog>
```

The fragment shows part of an event log stored in the so-called MXML format [24] that is used by ProM [22]. The MXML format is system-independent and using ProMimport it is possible to extract logs from a wide variety of systems, i.e., systems based on products such as SAP, Peoplesoft, Staffware, FLOWer, WebSphere, etc. and tailor-made systems.

The fragment shows that an MXML log contains information about one or more processes. Per process there can be multiple process instances (i.e., cases) as shown by the `ProcessInstance` element. In this example each instance refers to the treatment of a particular patient. Within each process instance there may be multiple `AuditTrailEntry` elements each referring to a particular event. Within each `AuditTrailEntry` element there can be multiple elements describing the nature and content of the event. The `WorkflowModelElement` element refers to some "model object", e.g., an activity or a subprocess. Note that in the MXML fragment some of the activity names shown in Figure 4 are used. The `EventType` element denotes the transactional property of an event. In the fragment we see `offer`, `start`, and `complete` as event types. MXML also supports event types such as `reassign`, `withdraw`, `autoskip`, `manualskip`, `suspend`, `resume`, etc. Note that this information is optional, i.e., in many logs only events of type `complete` are present because only the commit of a transaction in the system is logged. The elements `Timestamp`, `Originator`, and `Data` are also optional. The `Timestamp` element refers to the date and time of the event. If no timestamp is given, it is assumed that the event occurred in the order listed. The `Originator` element refers to the resource (i.e., person) executing the event. Using the `Data`

element it is possible to add arbitrary data to each event. The same element can be used to add data to process instances and processes.

Although we have a variety of real-life logs in MXML format (e.g., data of several hospitals), we use an artificially generated log in this paper. The reason is that it allows us the explain the concepts in a clear way. Real-life logs tend to require domain knowledge and typically refer to processes which are much larger and complex than the process shown in Figure 4.

We have used CPN Tools [21] to simulate the process and using ProMimport converted the simulation data of 1000 patients into MXML. Note that ProMimport allows for the collection and transformation of data from many systems including CPN Tools [33]. We will use this data in the remainder of this paper. The MXML fragment just discussed is part of this larger event log.

The MXML log will be used to discuss the functionality of ProM [22]. Since ProM has a good coverage of the 6 classes mentioned before, it is representative for what is possible with process mining today. Moreover, the staff assignment miner which we present in this paper has also been implemented in ProM and we will use the same event log to show the application of staff assignment mining.

2.3 Process Discovery

Process discovery refers to the various techniques for control-flow discovery. ProM currently offers eight plug-ins for discovering processes. These plug-ins use different "target formats", i.e., different languages to represent the result of the discovery algorithm (e.g., Petri nets, EPCs, heuristics nets).

Figure 5 shows four process models obtained using various plug-ins for control-flow discovery present in ProM. Figure 5(a) shows the result obtained by applying the α algorithm [11] to the log with the full transactional information, i.e., each activity characterized by three events in the log: `offer`, `start`, and `complete`. Hence, in the resulting Petri-net each activity is represented by three transitions. Figure 5(b) is also generated by the α algorithm but now by only considering the complete events. Now it is possible see that the α algorithm is able to discover the process depicted in Figure 4 without any a-priori information. Figure 5(c) shows the result of applying the multi-phase miner [23]. This approach first builds a model for every instance and then starts aggregating these instance models. The native format of the multi-phase miner is the EPC language. However, the multi-phase miner can also display its results as Petri nets. In fact, in ProM any Petri net can be converted into an EPC, YAWL model, or heuristics net and vice versa. Figure 5(d) shows a heuristics net discovered by the heuristics miner [59]. The format is also used by the genetic miner [6] and both are able to cope with noise.

2.4 Social Network Mining

Most related to the topic of this paper is the *social network miner* plug-in [7, 8]. Both the social network miner and the staff assignment miner aim to discover models related to the organization. While the staff assignment plug-in, presented

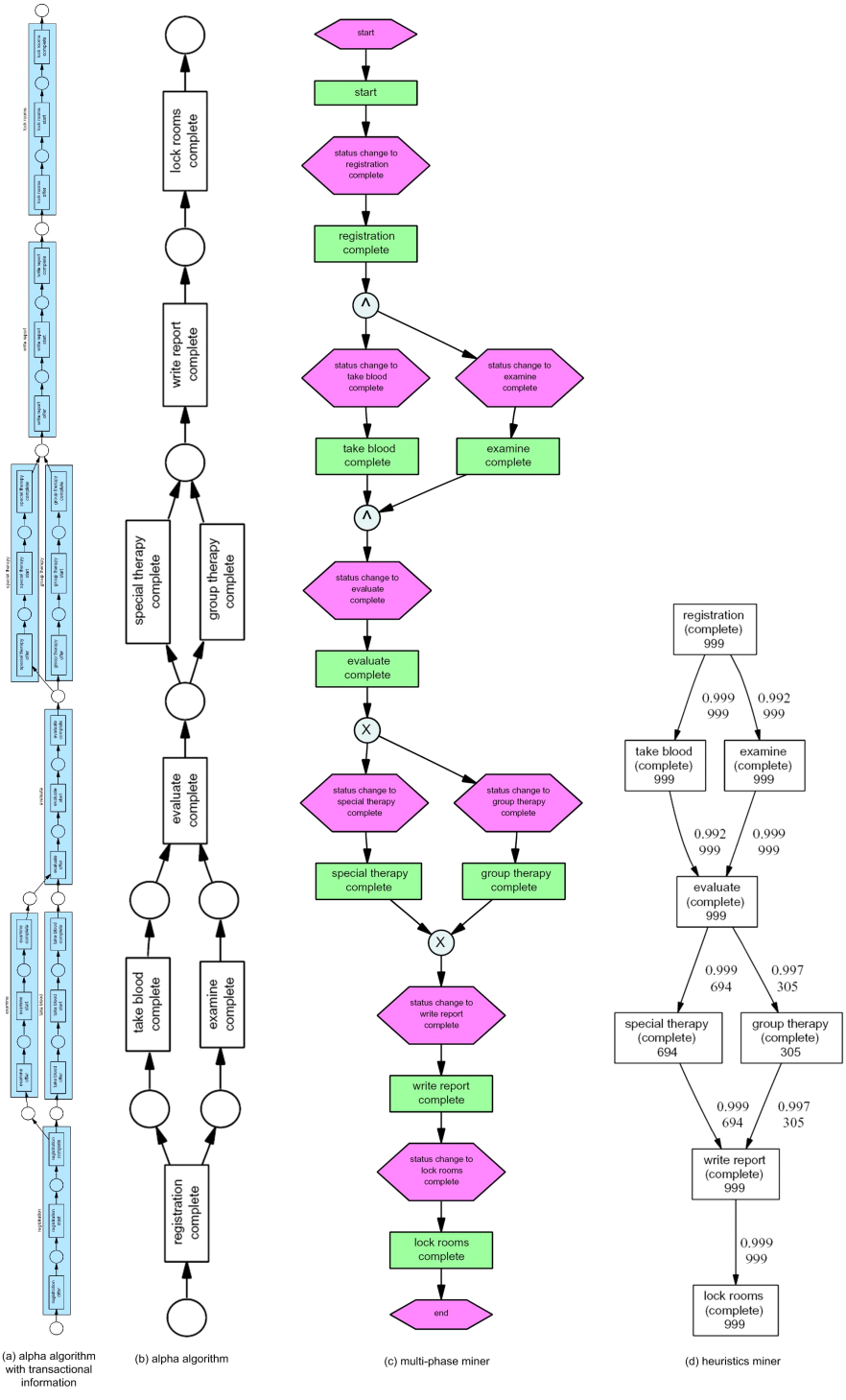


Fig. 5. Based on an analysis of an event log with data on 1000 patients, various mining plug-ins are able to discover the underlying process.

later in Section 3, assumes some a-priori information about the organization (e.g., roles and organization units), the social network miner looks at the information in the log only. The key information element in the log is the `Originator` element that refers to the resource (i.e., person/agent/individual/performer) executing the event. Using (1) metrics based on (possible) causality, (2) metrics based on joint cases, (3) metrics based on joint activities, and (4) metrics based on special event types, the social network miner attempts to discover the social relationships and their weight. Metrics based on (possible) causality monitor for individual cases how work moves among performers. One of the examples of such a metric is *handover of work*. Within a case (i.e., a process instance) there is a handover of work from individual i to individual j if there are two subsequent activities where the first is completed by i and the second by j. This notion can be refined in various ways as shown in [7, 8]. Metrics based on joint cases ignore causal dependencies but simply count how frequently two individuals are performing activities for the same case. If individuals work together on cases, they will have a stronger relation than individuals rarely working together. Metrics based on joint activities do not consider how individuals work together on shared cases but focus on the activities they perform. The assumption here is that people doing similar things have stronger relations than people doing completely different things. Each individual has a "profile" based on how frequent they conduct specific activities. There are many ways to measure the "distance" between two profiles thus enabling many metrics. Metrics based on special event types consider the event type. For example, if i frequently delegates work to j but not vice versa it is likely that i is in a hierarchical relation with j.

Based on these metrics one can determine if there is a social relationship between to individuals, and if so, the importance/weight of this relationship is established. This is then represented as a *social network* (i.e., a graph where each node represents an individual). Many notions have been defined for such networks [57, 53]. If all other individuals are in short distance to a given node and all geodesic paths (i.e., shortest path in the graph) visit this node, clearly the node is very central (like a spider in the web). There are different metrics for this intuitive notion of *centrality*. The Bavelas-Leavitt index of centrality is a well-known example that is based on the geodesic paths in the graph. Other related metrics are *closeness* (1 divided by the sum of all geodesic distances to a given resource) and *betweenness* (a ratio based on the number of geodesic paths visiting a given node). Another interesting metric is the *sociometric status* which is determined by the sum of input and output relations. It is also possible to determine *cliques*, i.e., groups of connected individuals with few relations to people outside this group.

Figure 6 shows a screenshot of the social network miner plug-in of ProM while analyzing the event log with data on 1000 patients. Using the "handover of work" a social network is built. The current social network miner plug-in of ProM offers little support for the analysis of social networks. Instead it provides exports to tools such as Agna and NetMiner.

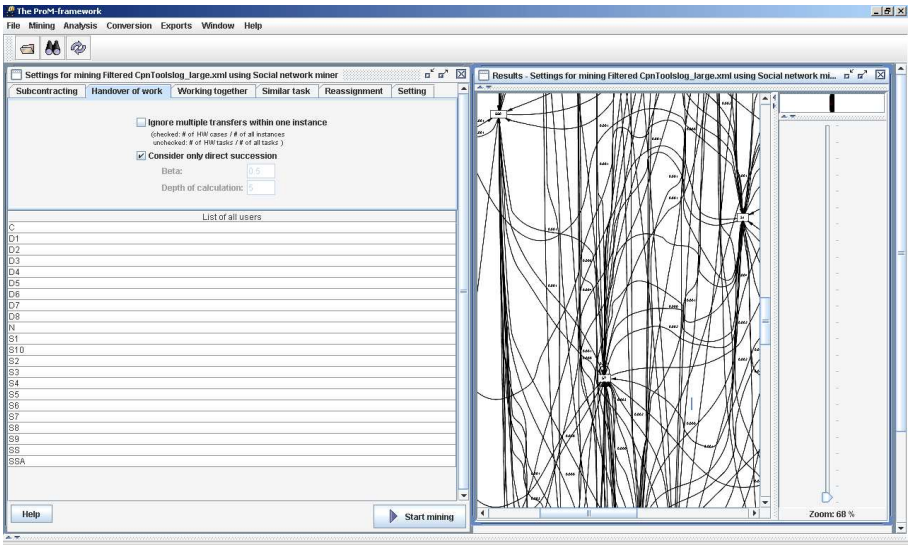


Fig. 6. Based on an analysis of an event log with data on 1000 patients, the social network miner constructs a social network that can be analyzed using dedicated SNA tools.

Role	Agent
Assistant for staff nurse	SSA
Deputy chief	SC
Doctor	D1, D2, D3, D4, D5, D6, D7
Head of department	C
Leader neurology	D1
Lead special therapy	SS
Night watchman	N
Nurse	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10
Nurse for night shift	S4, S7
Secretary	D8
Staff nurse	SS

Fig. 7. Organizational data containing agents and their assigned roles.

Before showing some results obtained using the social network miner plug-in and the social network analysis tool NetMiner, we return to our running example. In the MXML log the `originator` element refers to the person executing the corresponding activity, e.g., in the fragment shown before some person referred to as `S7` executes activity `registration` for process instance `567` (i.e., patient John Smith). In this paper, we will refer to `S7` as the agent performing the activity. Figure 7 shows the role of each agent. As shown, agent `S7` is both a `nurse` and a `nurse for night shift`. When we discuss the staff assignment rules, we will see that these roles are used for work distribution and authorization. For example, activity `take blood` can only be done by agents having the role `nurse` or the role `doctor`, e.g., nurse `S7` can take blood but the head of the department head (agent `C`) cannot. Figure 8 shows some screenshots of NetMiner while analyzing

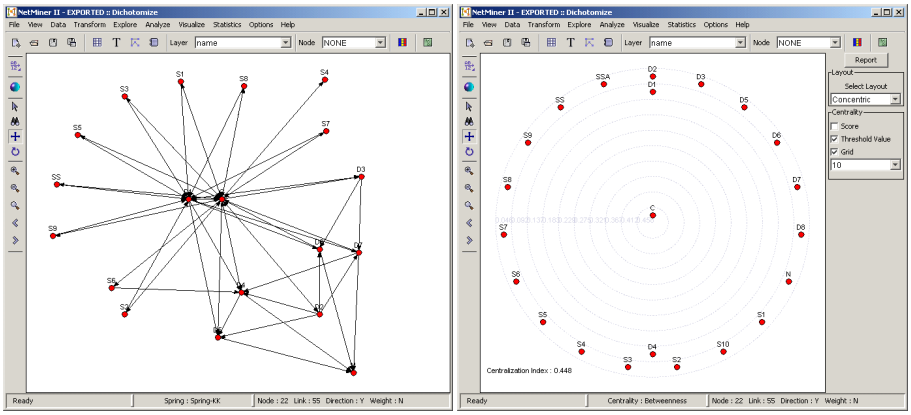
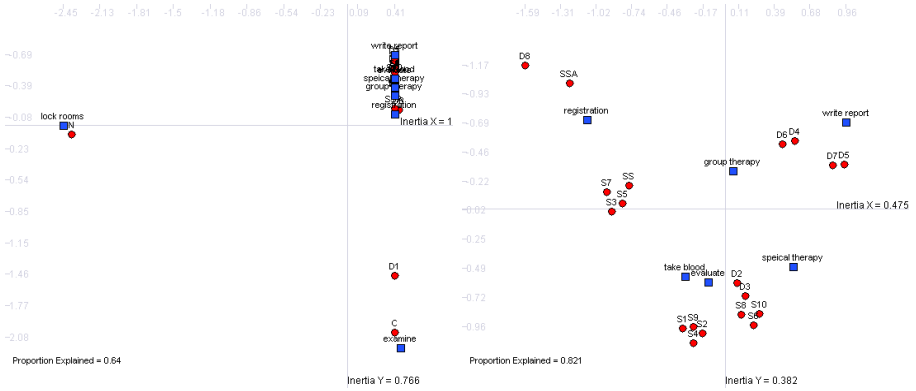


Fig. 8. Screenshots of NetMiner while analyzing the social network obtained in Figure 6.

the social network obtained using ProM. The screenshot on the left shows the social network when only considering the stronger relationships. (The two nodes in the middle are `C` and `D1`.) The screenshot on the right shows an analysis of "centrality" using the *betweenness* metric (as mentioned before, this ratio is based on the number of geodesic paths visiting a given node). This analysis clearly shows that the head of the department (agent `C`) is the most central person in the social network. As depicted, doctors `D1` and `D4` are more central than the other doctors. Doctor `D1` is also the leader of neurology which puts her more into the center. Doctor `D4` has the same role as the other doctors, but has additional abilities. In addition to roles (as shown in Figure 7) we will also consider *abilities* and *organizational units*. However, we will not elaborate on this now and refer to Section 3 instead.

Tools such as NetMiner offer many techniques to analyze the social network. For example, based on the MXML log we can analyze the frequencies of activities and use the hypothesis that actors with a similar "profile" are close in

the social network. Based on this we conduct a correspondence analysis [19] as shown in Figure 9. The screenshot on the left-hand side shows the result of correspondence analysis based on all actors and all activities. In the figure, boxes represent activities and circles represent agents. Closely positioned nodes (both activities and agents) indicate a strong correspondence while distant nodes are "more different". (Although the distance between both types of nodes should not be interpreted as an absolute measure [19].) In other words, in the two-dimensional space shown, similar nodes attract one another while dissimilar nodes repel one another. The left-hand side of Figure 9 shows that the agent `N` (the night watchman) is quite different from the other nodes and is only close to activity `lock rooms`. This is correct, he is the only person that can execute activity `lock rooms` and he is not allowed to execute any other activity. In the other dimension, agents `D1` and `C`, and activity `examine` are isolated from the rest of the nodes. Again this is a correct conclusion: the head of the department `C` and the leader of neurology `D1` are the only two that are allowed to execute activity `examine`. Note that correspondence analysis only analyses the actual execution profile of each actor and is not aware of any staff assignment rules. The screenshot on the right-hand side of Figure 9 shows the result of correspondence analysis after removing the activities and actors just mentioned, i.e., without agents `N`, `D1` and `C`, and activities `lock rooms` and `examine`. Again similar actors are grouped together and actors and activities that are related are positioned close to one another.



(a) With agents *N*, *D1*, and *C* and activities *examine* and *lock rooms*.

(b) Without agents *N*, *D1*, and *C* and activities *examine* and *lock rooms*.

Fig. 9. Using correspondence analysis to see how activities and agents fit together.

Although Figure 9 illustrates that correspondence analysis can be used to discover staff assignment rules, it is a rather crude mechanism that requires careful human interpretation. Hence, we will use *decision trees* instead (this will be explained in Section 3).

2.5 Decision Mining

After discussing plug-ins focusing on the control-flow and organizational perspectives, we now apply a plug-in focusing on the data perspective to the running example. This is the *decision miner*, i.e., a plug-in for the analysis of *decision points*. Note that using an discovery algorithm (e.g., the α algorithm [11]) it is possible to discover the control-flow perspective of a process. In such a discovered process there are decision points, e.g., in a Petri net places with multiple output arcs and in an EPC (X)OR-split connectors denote such decisions. The decision miner aims to discover the data influencing this decision using decision trees.

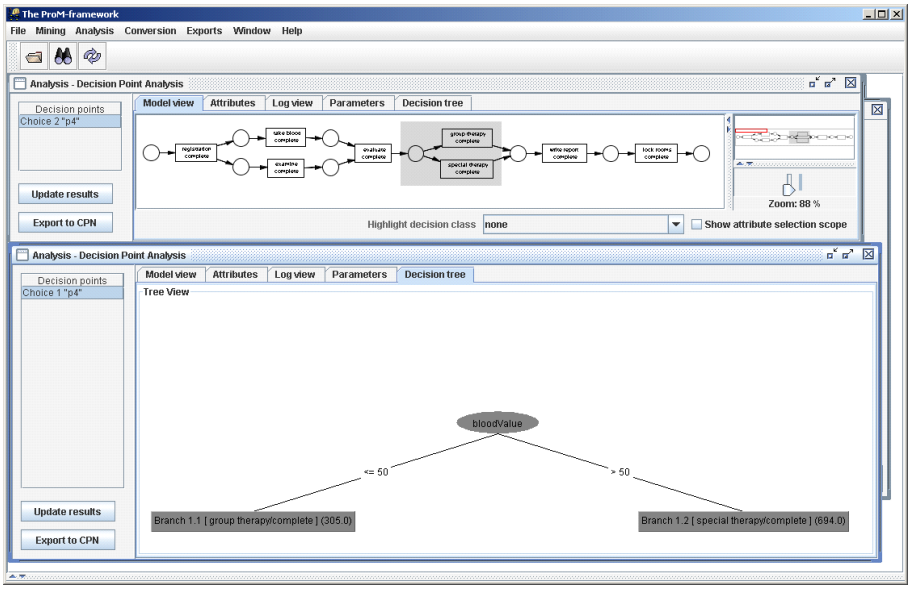


Fig. 10. Screenshots of the decision miner while analyzing the only decision point in the running process.

In the running example there is only one decision, i.e., the choice between `special therapy` and `group therapy`. This decision is made when completing activity `evaluate`. At this point in time, data such as the patient name, blood value, etc. are known. Clearly, the name of the patient is not relevant for the decision but the blood value is. This is discovered by the decision miner as shown in Figure 10. The two screenshots show that a decision point can be selected and analyzed. The result is depicted as a decision tree. In this example, the choice between `special therapy` and `group therapy` only depends on the data element `bloodValue` (with 50 being the split value).

2.6 Conformance Checking

So far the focus has been on discovery, i.e., discovering a process model, a social network, or a decision tree. To conclude this section we discuss two plug-ins that focus on conformance rather than discovery. The first plug-in we present is the *conformance checker*.

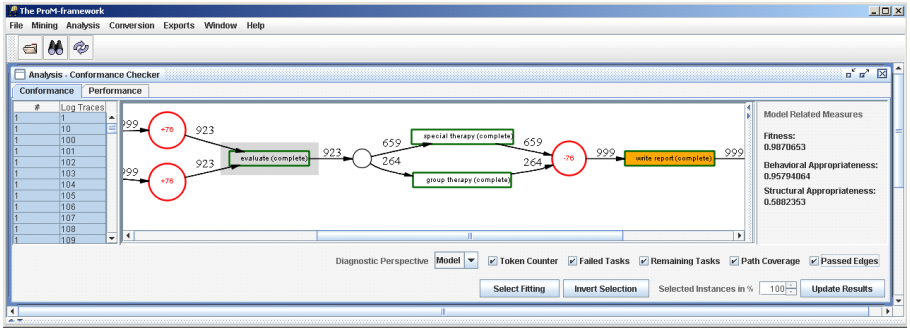


Fig. 11. Screenshot of the conformance checker.

To present we have created an event log containing some deviations. First of all, for some exceptional cases we inserted a new activity `lab test` that follows activity `examine`. Moreover, it is also possible that after taking the blood (activity `take blood`), the process immediately progresses to `write report` (again only for some exceptional cases). Now assume that the hospital considers the initial process as shown in Figure 4 to be the desirable process (i.e., the reference model). Using the conformance checker we can compare a real log with some a-priori model. Figure 11 shows the result. The screenshot shows the Petri net model of the initial process and highlights the points where the observed behavior deviates from the desirable behavior. When loading the conformance checker, a warning is given that activity `lab test` does not appear in the process model. After this warning, the conformance checker "replays" the history as recorded in the MXML log and notes the deviations related to the skipping of activities `evaluate` and `special/group therapy`. As shown in Figure 11, there were 76 cases where `evaluate` was supposed to be executed but instead directly `write report` was executed. Moreover, the conformance checker measures the fitness, structural appropriateness, and behavioral appropriateness. The fitness of the model is 0.98, i.e., 98 percent of the events in the log can be "explained" by the model. See [50] for more information about these metrics.

2.7 Property Checking

The conformance checker assumes some a-priori process model. In some cases, there is not a complete a-priori process model but merely a set of desirable or

undesirable properties. For example, "a request should always be followed by a response within a time period of 10 days", "under no circumstances both activities should be executed", or "two specific activities should never be executed by the same person". Such properties, also referred to as *business rules*, can be expressed in *temporal logic* [40]. The ProM framework provides an *LTL checker* that can check properties expressed in Linear Temporal Logic (LTL) [2]. LTL is a specific temporal logic and we have extended this on the basis of all possible information in MXML logs, i.e., in the LTL checker is is easy to refer to data, timestamps, agents, etc. Properties in the LTL checker can be parameterized and re-used, e.g., the 4-eyes principle ("two specific activities should never be executed by the same person") is defined once and can be re-used for any event log and serves as a basis for derived formulas (cf. [2]).

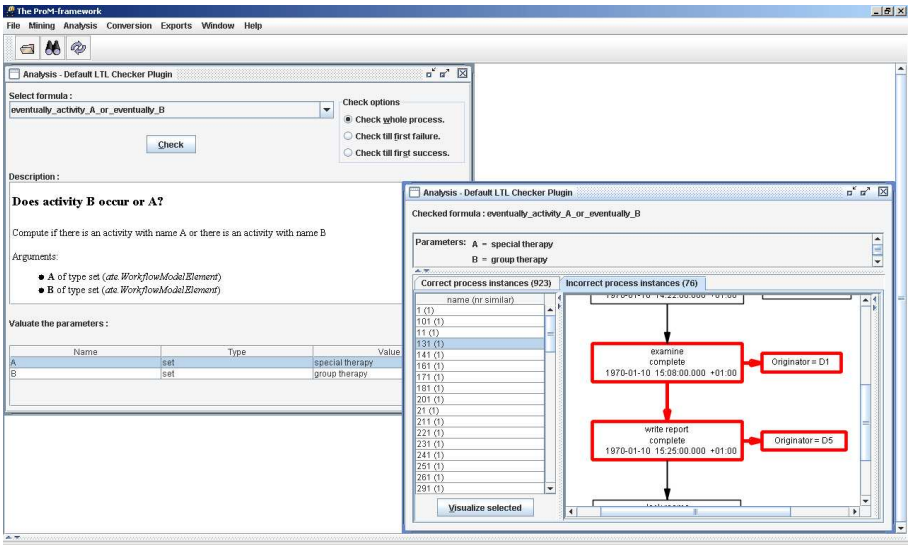


Fig. 12. Screenshot of the LTL checker plug-in.

Like the conformance checker the LTL checker aims at conformance rather than discovery. Therefore, we again use the event log with deviations. Using the LTL checker it is easy to find both types of deviations. Figure 12 shows the analysis of the property that each person should get therapy, i.e., at least one of the activities `group therapy` and `special therapy` should be executed for each patient. As Figure 12 shows, there are 76 cases where this is not the case and for patient 131 it is shown that indeed the activities `group therapy` and `special therapy` are skipped because `examine` is followed by `write report`.

The LTL checker can also be used to ask questions related to the organizational perspective. For example, we can investigate whether the 4-eyes principle applies to activities `examine` and `evaluate`. Using the LTL checker we discov-

ered that for 25 cases, both activities are executed by the same person. By again applying the LTL checker to these 25 cases we find that D1 is the only person violating the 4-eyes principle with respect to activities `examine` and `evaluate`. This shows that the LTL checker is a highly generic tool also useful for investigating staff assignment rules. However, using the LTL checker one can only check rules that have been explicitly formulated by the user of ProM. Using the `StaffAssignmentMiner` presented in the next section, no a-priori model/rule is needed (only the organizational context and the log are taken as input).

3 Mining Staff Assignment Rules

In this section we provide the basic definitions of organizational models and staff assignment rules (cf. Section 3.1). They are necessary for introducing the staff assignment mining approach in Section 3.2 as well as for the evaluation of the simulation results in Section 4.2.

3.1 Organizational Framework

In the following an (organizational) meta model is presented which is comparable to existing RBAC models (e.g., [27, 47]). It can be used for describing organizational entities and the relations between them (cf. Figure 13). In this paper we restrict our considerations to the basic entity types `Organizational Unit`, `Role`, `Ability`, and `Agent`, and to the particular relation types existing between them (e.g., agent `A1` belongs to organizational unit `O1`, role `R1` specializes role `R0`, etc.).

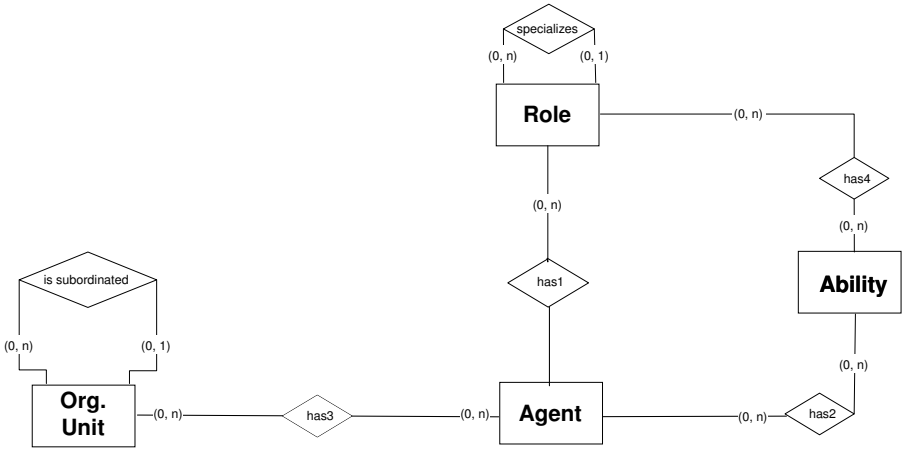


Fig. 13. Organizational meta model

Based on this meta model, we formalize the notion of *organizational models*. We have also developed an XML format for storing such models. Since the focus is on staff assignment mining rather than on organizational modeling, we do not consider the cardinalities depicted in Figure 13.

Definition 1 (Organizational Model). *An organizational model is a tuple* $OM = (Agents, Roles, Abilities, OrgUnits, has1, has2, has3, has4, is\_subordinated, specializes)$*, where:*

- $Agents$ *is the set of agents (i.e., the people performing activities),*
- $Roles$ *is the set of roles,*
- $Abilities$ *is the set of abilities,*
- $OrgUnits$ *is the set of organizational units,*
- $has1 \subseteq Agents \times Roles$ *is the relation linking agents to roles,*
- $has2 \subseteq Agents \times Abilities$ *is the relation linking agents to abilities,*
- $has3 \subseteq Agents \times OrgUnits$ *is the relation linking agents to organizational units,*
- $has4 \subseteq Abilities \times Roles$ *is the relation linking abilities to roles,*
- $is\_subordinated \subseteq OrgUnits \times OrgUnits$ *defines the organizational hierarchy, and*
- $specializes \subseteq Roles \times Roles$ *defines the role hierarchy.*

As part of the organizational model used in our example we have already introduced the set of agents and their assigned roles in Figure 7. Additionally, the agents may belong to organizational units (e.g., agents `SC`, `C`, and `N` belong to organizational unit `hospital`, cf. Figure 14). The roles can be specialized (e.g., a `staff nurse` is a `nurse`) and the organizational units can be subordinated to other organizational units (e.g., organizational unit `administration` is subordinated to organizational unit `hospital`) as depicted in Figure 14. Due to space restrictions we omit the associations between agents and abilities. Moreover, for simplicity reasons we ignore relation *has4*. (This relation has only been added to explicitly check the consistency between the roles and abilities of an agent.)

OrgUnit	Agent
Administration	SSA, C, SC, N, D8
Hospital	SC, C, N
Neurology	D1, D6
Special therapy	S6, S8, S10, D2, D3
Therapy	SS, SSA, S1, S2, S3, S4, S5, S7, S8, S9, D4, D5, D6, D7

OrgUnit	Subordinated OrgUnits
Hospital	Administration, Special therapy, Therapy

Role	Specialized Role
Nurse	Staff nurse

Fig. 14. Organizational data containing organizational units and belonging agents, specialization relation between roles, and subordinated organizational units.

To be able to use the relations mentioned in Definition 1, we define the following basic notations.

Definition 2 (Relations and Functions). *Let U be some universe of discourse and $R \subseteq U \times U$ some relation. For any $u \in U$: $R(u) = \{x \in U \mid (u, x) \in R\}$. R^* is the transitive closure of R. Let $f \in A \rightarrow B$ be a function with domain A and range B. For any set $X \subseteq A$: $f(X) = \{f(x) \mid x \in X\}$.*

These notations can be applied to the relations of Figure 7, e.g., *has1*(`staff nurse`) is the set of agents with role `staff nurse`. *is_subordinated** is the transitive closure of the organizational hierarchy. *has3*(`hospital`) is the set of agents in hospital. These are the agents `SC`, `C`, and `N`. However, the organizational unit `hospital` contains other units. Using *has3*(*is_subordinated**(`hospital`)) we obtain people that are indirectly associated to the organizational unit `hospital`, e.g., the agents in `therapy`. Using *has1*(*specializes**(`nurse`)) we obtain people that are indirectly associated to the role `nurse`, i.e., also people having role `staff nurse`.

Based on the organizational entities and relations described by *OM* we can define *staff assignment rules* in order to specify the assignment of agents to process activities. Since the structuring and semantics of the staff assignment rules is fundamental for the (semi-)automated derivation of rule adaptations, we consider this issue in more detail. We distinguish between elementary and complex staff assignment rules.

Definition 3 (Elementary Staff Assignment Rule). *Let OM be an organizational model as defined in Definition 1. An elementary staff assignment rule* `EAR` *on OM is defined as follows:*
`EAR` $\equiv$ `(EAR1` $\longleftarrow$ `(Role =` r`))` | `(EAR2` $\longleftarrow$ `(Ability =` a`))` |
`(EAR3` $\longleftarrow$ `(OrgUnit =` o`))` | `(EAR4` $\longleftarrow$ `(Role+ =` r`))` |
`(EAR5` $\longleftarrow$ `(OrgUnit+ =` o`))` | `(EAR6` $\longleftarrow$ `(Agent =` ag`))`.
For each possible elementary staff assignment rule we define the set of all agents that qualify (we call this the valid agent set or VAS for short):

- $VAS(OM, \texttt{EAR1}) = has1(r)$ *is the set of agents having role* r,
- $VAS(OM, \texttt{EAR2}) = has2(a)$ *is the set of agents having ability* a,
- $VAS(OM, \texttt{EAR3}) = has3(o)$ *is the set of agents in organizational unit* o,
- $VAS(OM, \texttt{EAR4}) = has1(specializes^*(r))$ *is the set of agents (indirectly) having role* r, *and*
- $VAS(OM, \texttt{EAR5}) = has3(is\_subordinated^*(o))$ *is the set of agents (indirectly) in organizational unit* o
- $VAS(OM, \texttt{EAR6}) = \{ag\}$.

In order to enable the definition of more complex staff assignment rules we allow for the composition of existing rules (cf. Definition 4). For this purpose the following operators can be used: negation, conjunction, and disjunction.

Definition 4 (Staff Assignment Rule). *Let OM be an organizational model (cf. Definition 1). A staff assignment rule* `AR` *is defined recursively:*
`AR` $\equiv$ `EAR` | `NAR` | `CAR` | `DAR`, *where*

– `EAR` *is an elementary staff assignment rule (cf. Definition 3),*
– `NAR` ⟵ `(NOT (AR))` *where* `AR` *is a staff assignment rule,*
– `CAR` ⟵ `(AR1 AND AR2)` *with* `AR1` *and* `AR2` *are staff assignment rules, and*
– `DAR` ⟵ `(AR1 OR AR2)` *with* `AR1` *and* `AR2` *are staff assignment rules.*

For each possible elementary staff assignment rule we define the set of all agents that qualify:

– $VAS(OM, \texttt{EAR})$ *is defined in Definition 3,*
– $VAS(OM, \texttt{NAR}) = Agents \setminus VAS(OM, \texttt{AR})$ *is the set of agents not qualifying for* `AR`*,*
– $VAS(OM, \texttt{CAR}) = VAS(OM, \texttt{AR1}) \cap VAS(OM, \texttt{AR2})$ *is the set of agents qualifying for* `AR1` *and* `AR2`*,*
– $VAS(OM, \texttt{DAR}) = VAS(OM, \texttt{AR1}) \cup VAS(OM, \texttt{AR2})$ *is the set of agents qualifying for* `AR1` *or* `AR2`*.*

Definitions 3 and 4 define the set of agents qualifying for any staff assignment rule. Given a staff assignment rule `AR`, $VAS(OM, \texttt{AR})$ is the set of qualifying agents.

For more details on the framework for defining and changing organizational models and staff assignment rules see [47].

3.2 Decision Tree Learning

In this section we provide background information on staff assignment mining based on decision tree learning as introduced in [39]. Since staff assignment rules are supposed to identify the set of real performers of a given activity x, the challenge is to determine combinations of properties that distinguish performers from non-performers. Thus, the problem of deriving the rules can be interpreted as an inductive learning task from positive and negative examples.[2] Unlike with *control-flow mining* negative examples are directly given for our problem: every non-performer can serve as a negative example. First, we define the notion of positive and negative examples for this learning problem.

Definition 5 (Positive/Negative Examples). *Let A be a set of agents and let X be the total set of activities. Then performer is a classification function which determines whether a given agent $a \in A$ has worked on any instance of activity $x \in X$ or not: performer* : $A \times X \rightarrow \{\texttt{True}, \texttt{False}\}$

$$performer(a, x) = \begin{cases} \texttt{True} & \textit{if } a \textit{ has performed an instance of } x \\ \texttt{False} & \textit{otherwise} \end{cases}$$

The triple $(x, a, performer(a, x))$ represents an "example". We further distinguish between positive examples, i.e., $(x, a, \texttt{True})$, and negative examples, i.e.,

[2] Note that we need to assume some notion of completeness, i.e., if an agent is able to perform an activity, it will be observed. However, this notion of completeness seems to be realistic and is much weaker than the notions of completeness typically used for control-flow mining.

$(x, a, \texttt{False})$. *Note that due to this definition, agents performing x multiple times will be associated with a respective number of examples. For every non-performer a negative example can be generated.*

Based on the examples the objective is to derive a hypothesis $h(a)$ which approximates the classification function $performer(a)$. This problem belongs to *supervised learning* [34] since we have predefined classes. Many learning methods can be applied to solve this problem. We have chosen to adapt *decision tree learning* [18].

Decision tree learning is one of the most widely-used methods of inductive inference. It can be employed for attribute-based learning of disjunctive concepts. This method is simple and explicitly enables graphical representations. This constitutes an advantage when developing a user-friendly graphical interface for a respective staff assignment mining tool. Furthermore, decision tree learning also incorporates methods for handling noise data and continuous attribute values.

All entities of an organizational model can be used as testing attributes in order to separate the performers from the non-performers of a given task x.

Staff assignment rules can be derived from building decision trees. Starting at the root an attribute is chosen in order to separate the example set. Which attributes are chosen and in which order is discussed in the following. This procedure is continued recursively for the child nodes until only examples from one class (indicated by the '+' and the $-$' set are left or no attributes are left). The '+' set represents the class of performers while the '$-$' set represents the class of non-performers. Depending on whether they are related to an organizational entity, examples (i.e., agents) are assigned to the "yes"-child-node or "no"-child-node respectively. Note that for every agent it can be determined whether the agent is related to an organizational entity or not. From a decision tree if-then-rules or rules in disjunctive normal form (DNF) can be easily derived. The conjunction of attribute values of a path from a leaf-node with the target class to the root represents the if-part of the if-then rule or a disjunction element of the DNF.

However, our objective is to mine general profiles of performers with as few conjunction elements as possible. Finding decision trees representing minimal rules is of NP-hard complexity [44]. For guiding the search, i.e. chosing an attribute, the metric *information gain* [45, 44] is used. The *information gain* metrics is based on entropy calculations. The formulas for *entropy* and *information gain* are given below. S is an example set, a an attribute, and p_+ and p_- indicate the proportion of positive and negative examples respectively. S_{yes} and S_{no} are the example sets assigned to the "yes"- or the "no"-child of the node belonging to S, respectively.
The entropy is a metrics for the homogeneity of a set. At every separation step the attribute with the best *information gain* value is chosen. Thus, the decision tree algorithm tries to achieve the best split in every step.

$$entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \tag{1}$$

$$information\ gain(S,a) = entropy(S) - \frac{|S_{yes}|}{|S|} entropy(S_{yes}) - \frac{|S_{no}|}{|S|} entropy(S_{no}) \quad (2)$$

For further information on decision trees and metrics please refer to [45, 44, 41].

Generally more than one decision tree can be derived. Therefore, it is important to offer alternative rules to the process engineer who then evaluates them. In order to extract more than one rule backtracking is needed. Again, *information gain* can be used in order to chose the suitable attributes. Instead of using only the best separating attribute the k-best attributes can be used, where k is a configurable parameter.

3.3 Implementation within ProM

The staff assignment mining approach based on decision tree learning is realized as plug-in of the ProM framework (called `StaffAssignmentMiner`). When this plug-in is started it imports an organizational model (represented as XML file) and process execution logs represented within the ProM MXML audit trail format [24]. After the import users can specify certain parameters for the mining process like the activity for which the staff assignment rule is to be mined as well as the number of decision trees to be generated (e.g., parameter "k best attributes", cf. Figure 15). The decision trees and the resulting staff assignment rules are visualized as depicted in Figure 16. For an a-posteriori analysis, it is possible to specify thresholds for the number of performers / non-performers in order to prune the decision trees accordingly (cf. Figure 15).

4 Diagnosis and Optimization of Staff Assignment Rules

Using the `StaffAssignmentMiner` plug-in of the ProM framework, the "real" staff assignment rules (which are reflected by the execution logs) can be mined and compared with the staff assignment rules defined for the underlying process afterwards. Based on this comparison, possible deviations between existing and mined staff assignment rules can be automatically detected (we call this staff assignment rules diagnosis). Section 4.1 illustrates the diagnosis phase based on the simulation data for the patient treatment process (cf. Figure 4).

However, the results of comparing existing and mined staff assignment rules have to be interpreted afterwards. The goals are 1) to either comfirm the existing staff assignment rules or to come up with suggestions for improving them and 2) to learn more about the work practice of the analyzed organization (e.g., to learn about substitution behavior among colleagues). Section 4.1, first of all, motivates the different relations between existing and mined staff assignment rules by presenting the comparison results for the patient treatment process.

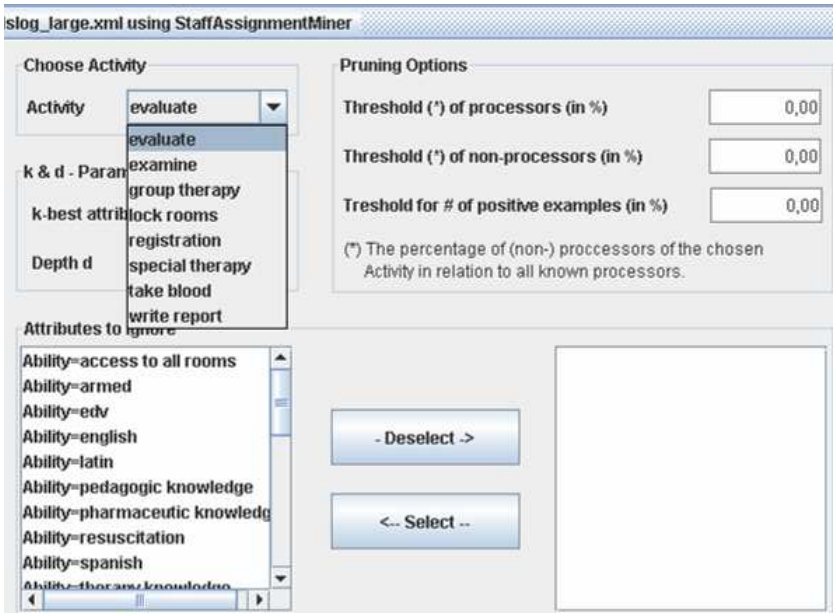


Fig. 15. Staff assignment plug-in for the ProM framework.

Then, in Section 4.2, the possible syntactical relations between existing and mined staff assignment rules are formalized to provide a basis for an automatic comparison. How these syntactical relations can be interpreted is presented in Section 4.3. The whole approach is finally extended with respect to exceptional agent behavior in Section 4.4.

4.1 Simulation-Based Analysis of Staff Assignment Mining

Table 1 summarizes the original and mined staff assignment rules for all activities of our example treatment process (cf. Figure 4). In order to be able to compare the original and the mined staff assignment rules we logically remove those parts from the mined staff assignment rules which are built by redundant `FALSE` branches in the decision tree (i.e., the valid agent set for the staff assignment rule does not change when removing the part associated with the `FALSE` branch). For example, for activity `examine` staff assignment mining yields the following rule:

$SAR_{mined}^{examine}$ ⟵ `((Role="head of department") OR (Role="leader neurology") AND NOT (Role="head of department")))`.

Since the valid agent sets of rules `EAR1` ⟵ `(Role="head of department")` and `EAR2` ⟵ `(Role="leader neurology")` are disjoint over the underlying organizational model, rule `EAR3` ⟵ `(NOT (Role="head of department"))` is redundant. Therefore we (logically) remove these parts from the mined staff assignment rules in order to provide a suitable basis for the following evaluation.

In the remainder of this section we try to conclude certain relations between original and mined staff assignment rules which are then formalized in Section 4.2. These relations build the basis for suggesting optimizations for the predefined staff assignment rules in the following.

Table 1. *Original and resulting staff assignment rules after simulation.*

Activity x, original and mined staff assignment rules $\mathbf{SAR}^{x}_{original}$ / $\mathbf{SAR}^{x}_{mined}$
x = registration
$SAR^{registration}_{original}$ ⟵ ((Role="secretary") OR ((Ability="edv") AND (Ability="english")))
$SAR^{registration}_{mined}$ ⟵ ((Ability="edv") AND (Ability="english"))
x = examine
$SAR^{examine}_{original}$ ⟵ ((Role="head of department") OR (Role="leader neurology"))
$SAR^{examine}_{mined}$ ⟵ ((Role="head of department") OR (Role="leader neurology"))
x = take blood
$SAR^{takeblood}_{original}$ ⟵ ((Role="doctor") OR (Role="nurse"))
$SAR^{takeblood}_{mined}$ ⟵ (NOT (Role="assistant for staff nurse")) AND (NOT (OrgUnit="administration"))
x = evaluate
$SAR^{evaluate}_{original}$ ⟵ ((Role="doctor") OR (Role="nurse"))
$SAR^{evaluate}_{mined}$ ⟵ (NOT ((Role="assistant for staff nurse") AND (NOT (OrgUnit="administration")))
x = special therapy
$SAR^{specialtherapy}_{original}$ ⟵ ((Role="doctor") OR ((Role="nurse") AND (OrgUnit="special therapy")))
$SAR^{specialtherapy}_{mined}$ ⟵ ((Role="doctor") OR (NOT (Role="doctor") AND (OrgUnit="special therapy")))
x = group therapy
$SAR^{grouptherapy}_{original}$ ⟵ ((Role="doctor") OR Role=("assistant for staff nurse") OR (Role="staff nurse"))
$SAR^{grouptherapy}_{mined}$ ⟵ ((Role="doctor") OR (Role="assistant for staff nurse") OR (NOT (Role="doctor") AND (Ability="pharmaceutic knowledge") AND (Ability="edv")))
x = write report
$SAR^{writereport}_{original}$ ⟵ ((Role="doctor") OR (OrgUnit="therapy"))
$SAR^{writereport}_{mined}$ ⟵ ((Role="doctor") OR (OrgUnit="therapy"))
x = lock rooms
$SAR^{lockrooms}_{original}$ ⟵ (Role="night watchman")
$SAR^{lockrooms}_{mined}$ ⟵ (Role="night watchman")

From Table 1 we can see that for some activities the mined staff assignment rule exactly matches the original one (this holds for activities `examine`, `write report`, and `lock rooms`). Apparently, an exact match between original and mined staff assignment rule is more likely if only a small percentage of all agents qualifies for them (for activities `examine`, `write report`, and `lock rooms` the percentage of qualifying agents ranges from 4% and 17%).

For other activities the mined staff assignment rules differ from the orginal ones. This is, for example, the case for activity `registration` as depicted in Figure 16. The first separating attribute is given by ability `edv`, the second one

by ability `english`. The resulting staff assignment rule is

$\texttt{SAR}^{registration}_{mined} \longleftarrow$ `((Ability="edv") AND (Ability="english"))`

which corresponds to the second part of the original staff assignment rule (cf. Table 1). The other part (i.e., `Role="secretary"`), however, is redundant since it does not influence the valid agent set. Therefore it is not taken into account by the mining algorithm, even if we select more than $k = 1$ best attributes for analysis. In this case the staff assignment mining filters out *redundant* parts of the staff assignment rule and therefore *refines* the original staff assignment rule.

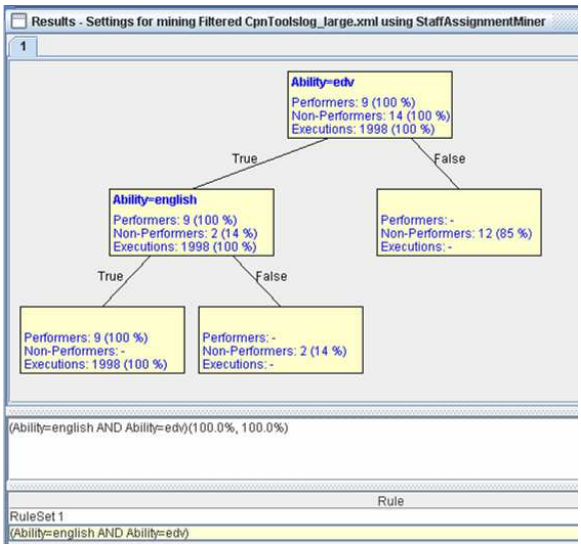


Fig. 16. Decision tree and staff assignment rule for activity `registration`.

For the remaining activities staff assignment mining results in so-called *complementary* staff assignment rules; i.e., the valid agent set for the original and for the mined staff assignment rules are the same. However, the mined staff assignment rules are (partially) build up by negating (elementary) staff assignment rules of the original rules. One example is the mined staff assignment rule for activity *evaluate* (cf. Figure 17). The resulting staff assignment rule

$\texttt{SAR}^{evaluate}_{mined} \longleftarrow$ `(NOT (Role="assistant for staff nurse")) AND`
`(NOT (OrgUnit="administration"))`

excludes all agents which do not have role `doctor` or role `nurse`. This, however, exactly corresponds to the negation of original staff assignment rule

$\texttt{SAR}^{evaluate}_{original} \longleftarrow$ `((Role="nurse") OR (Role="doctor"))`.

When analzing simulation data we observe that complementary staff assignment rules are derived if a high percentage of all agents qualifies for the original staff

assignment rule (i.e., 78, 3% for
$\texttt{SAR}^{evaluate}_{original} \longleftarrow$ `((Role="nurse") or (Role="doctor"))`).

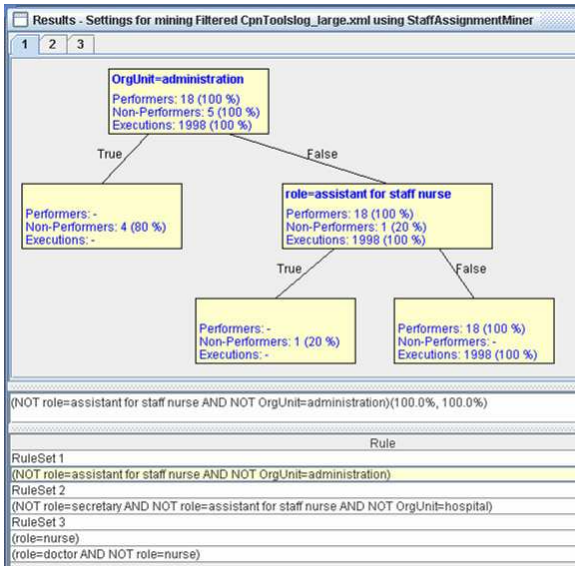


Fig. 17. Mining results for activity `evaluate`.

4.2 Syntactical Comparison of Staff Assignment Rules

In the previous section, different relations between original and mined staff assignment rules have been observed, comprising exact match, refinement, and complement. In this section, we provide a formalization based on the valid agent sets of original and mined staff assignment rules (cf. Definitions 3 and 4). In general, the following relations may occur as depicted in Figure 18. First of all, the valid agent sets may be equal for original and mined staff assignment rule (a). We denote this as *agent match*. In case (b) the mined staff assignment rule specifies a subset of the valid agent set of the original rules. Since only a subset of the agents qualifying for the original staff assignment rule actually works on the affected activity, this may indicate some kind of refinement (therefore we denote this as *agent refinement*). Consequently, the mining result can be suggested as optimization for the pre-defined rule. Contrary, the mined staff assignment rules may also specify a superset of the valid agent sets of the original staff assignment rules (i.e.,
$VAS(\texttt{OM}, \texttt{SAR}_{original}) \subset VAS(\texttt{OM}, \texttt{SAR}_{mined})$, cf. Figure 18(c)). This happens if agents have exceptionally worked on the affected activity, for example, as substitutes. This case is denoted as *agent expansion*. Case (d) (called *agent mismatch*

with overlap) constitutes a mixture of refinement as in case (b) and expansion as in case (c). Finally, for case (e) none of the qualifying agents has worked on the related activity and therefore the valid agent sets of original and mined staff assignment rule are disjoint (therefore we denote this case as *agent mismatch without overlap*).

The interesting question is how the mining results together with the relations between original and mined staff assignment rules based on their valid agent sets above can be used for optimizing the original rules.

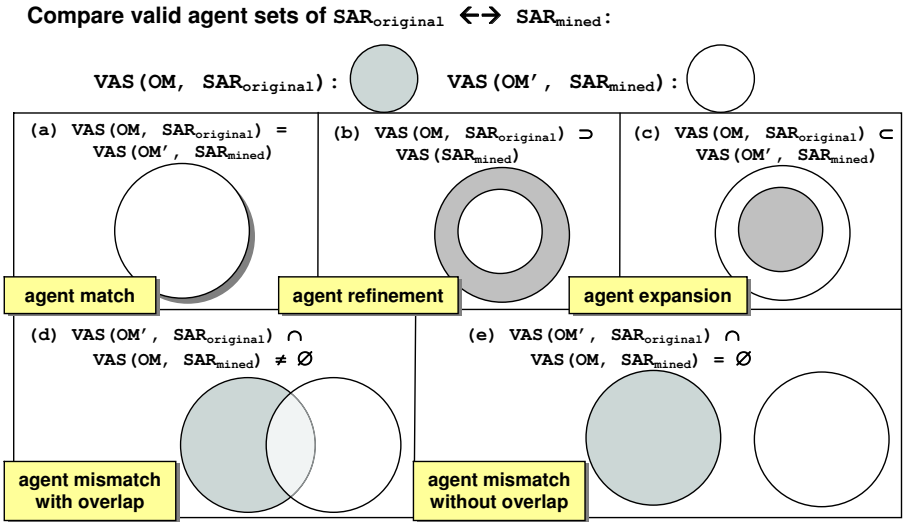


Fig. 18. Possible relations between valid agent sets of original and mined staff assignment rules.

4.3 Evaluation Framework

In this section we interpret the mining results in case of agent match, agent refinement, and agent expansion (cf. Figure 18 (a), (b), and (c)). As it can be seen from the following examples, often a pure agent set based evaluation is too vague. We obtain, for example, an agent match for activities `registration`, `examine`, and `evaluate`. However, the interpretation of having an agent match is different for each activity. First of all, for activity `registration`

$VAS(\texttt{OM}, \texttt{SAR}^{registration}_{mined}) = VAS(\texttt{OM}, \texttt{SAR}^{registration}_{original}) =$

$\{\texttt{SS}, \texttt{SSA}, \texttt{S3}, \texttt{S5}, \texttt{S7}, \texttt{S10}, \texttt{D4}, \texttt{D6}, \texttt{D8}\}$ holds (corresponding to the number of performers depicted in the associated decision tree, cf. Figure 16). As discussed in Section 4.1 the mined staff assignment rule itself "refines" the original one. Reason for this is that within the original staff assignment rule elementary rule

$\texttt{EAR} \longleftarrow \texttt{(Role="secretary")}$ is redundant since it does not provide any further seperation information (compared to the mined staff assignment rule). By contrast, for activity `examine` not only the agent sets are equal, also the original and mined staff assignment rules completely match. Finally, for activity `evaluate` also the valid agent sets match but the mined staff assignment rule seems to be a complement of the original one (cf. Section 4.1). By summarizing these observations we obtain the following sub-cases and related intepretations for an agent match between original and mined staff assignment rules for a certain activity x:

(a) Agent match (i.e., $VAS(\texttt{OM}, \texttt{SAR}^x_{mined}) = VAS(\texttt{OM}, \texttt{SAR}^x_{original})$):

- A *rule match* (i.e., $\texttt{SAR}^x_{mined} \equiv \texttt{SAR}^x_{original}$) is on hand if all elementary staff assignment rules and all logical connectors between the (elementary) staff assignment rules contained in $\texttt{SAR}^x_{mined}$ and $\texttt{SAR}^x_{original}$ are equal (only the order between the contained staff assignment rules may differ).
  Evaluation: Apparently, a rule match confirms the original staff assignment rule since during the actual process executions agents only work on activities they are qualifying for according to the original staff assignment rule. Therefore the quality of the original staff assignment rule has been proven by the staff assignment mining and does not have to be adapted for further process execution.
- A *rule refinement* between $\texttt{SAR}^x_{mined}$ and $\texttt{SAR}^x_{original}$ is on hand if $\texttt{SAR}^x_{original}$ equals a concatenation of $\texttt{SAR}^x_{mined}$ and some additional rule `SR` in a disjunctive way (i.e., $\texttt{SAR}^x_{original} \equiv \texttt{SAR}^x_{mined}$ `OR` `SR`[3]).
  Evaluation: $\texttt{SAR}^x_{mined}$ provides the same information as the original staff assignment rule does. Thus the original staff assignment rule can be replaced by the mined staff assignment rule. One advantage of doing so is that the redundant parts do not longer have to be evaluated when the staff assignment rule is resolved. For a PAIS controlling thousands of running instances this can be a noticeable optimization.
- In all other cases we obtain a total or partial *rule complement*.
  Evaluation: A mined staff assignment rule which is complementary to the original one may indicate that a high percentage of all agents qualifies for the original staff assignment rule. The evaluation of this (i.e., whether the original staff assignment rule is maybe too unspecific) has to be done by the user. In any case, the original staff assignment rule must not be replaced by the mined staff assignment rule automatically since this might lead to incorrect staff assignment resolutions afterwards. Assume, for example, that the `assistant for staff nurse` is reassigned from organizational unit `administration` to organizational unit `therapy`. Wrongly, the assistant would then qualify for activity `evaluation` in the sequel.

(b) Agent refinement (i.e., $VAS(\texttt{OM}, \texttt{SAR}^x_{mined}) \subset VAS(\texttt{OM}, \texttt{SAR}^x_{original})$):
Evaluation: In this case only agents contained in $VAS(\texttt{OM}, \texttt{SAR}^x_{mined})$ have actually performed activity x. Thus mined rule $\texttt{SAR}^x_{mined}$ is more specific than

[3] Maybe some reordering of $\texttt{SAR}^x_{original}$ or $\texttt{SAR}^x_{mined}$ is necessary in advance.

$\texttt{SAR}^{x}_{original}$ and may therefore replace the original rule (after being checked by, for example, the process designer).

Table 2 evaluates the relations between the original and the mined staff assignment rules based on the basic simulation data and the interpretation possibilities presented above (note that all original and mined staff assignment rules match with respect to their valid agent set). Furthermore, some optimization suggestions based on the analysis results are provided.

(c) Agent expansion (i.e., $VAS(\texttt{OM, SAR}^{x}_{original}) \subset VAS(\texttt{OM, SAR}^{x}_{mined})$): *Evaluation:* It is possible to suggest substitution rule SR based on staff assignment mining if the mining result $\texttt{SAR}^{x}_{mined}$ equals a concatenation of original rule $\texttt{SAR}^{x}_{original}$ and additional rule SR in a disjunctive way
(i.e., $\texttt{SAR}^{x}_{mined} \equiv \texttt{SAR}^{x}_{original}$ OR SR). Then the valid agent set of SR comprises all agents which are contained in the difference set between the valid agent sets of mined and original staff assignment rule
(i.e., $VAS(\texttt{OM, SR}) = VAS(\texttt{OM, SAR}^{x}_{mined}) \setminus VAS(\texttt{OM, SAR}^{x}_{original})$).

In order to illustrate the evaluation of case (c) we have to extend the basic simulation towards exceptional agent behavior data. The respective scenarios and the resulting substitution rules are presented in Section 4.4.

For cases (d) and (e) depicted in Figure 18 there are significant deviations regarding the set of agents having worked on the analyzed activity from the specified one. This fact is reflected by mined staff assignment rules strongly or even totally differing from the original ones. In this case, often it is reasonable to suggest a replacement of the original staff assignment rules by the mined rules (possibly after evaluation by the process designer).

4.4 Mining Deviations from Staff Assignment Rules

As already mentioned deviations from the specified staff assignment rules usually occur in practice, for example, if an agent is substituted by a colleague who actually does not qualify for the associated rule (e.g., due to vacation, disease, etc). It would be helpful if such substitution rules which have not been specified so far but are reflected within the logs could be detected using staff assignment mining. In order to analyze this question we adapt our simulation scenario as follows:

1. For activity `examine` with original staff assignment rule
   $\texttt{SAR}^{examine}_{original} \longleftarrow$ `((Role="head of department") OR (Role="leader neurology"))`
   we include a substitution by agent D4 (i.e., by adding rule R1 $\longleftarrow$ `(Agent="D4")`) for 5% of the executed instances. Practical background could be that in some cases the head of the department as well as the leader of the neurology are busy such that one of the doctors (i.e., D4) takes over to examine the patient.
2. Exceptionally, the report after therapy (activity `write report` with
   $\texttt{SAR}^{writereport}_{original} \longleftarrow$ `((Role="doctor") AND (OrgUnit="therapy"))`)
   might be also written by the staff nurse (i.e., we add rule
   R2 $\longleftarrow$ `(Role="staff nurse")` for 5% of the simulated instances).

Table 2. *Evaluation of the simulation results*

Activity x	**Relation between $\mathbf{SAR}^{x}_{original}$ and $\mathbf{SAR}^{x}_{mined}$** (note that ∀ x: actor match between $\mathrm{SAR}^{x}_{original}$ and SAR^{x}_{mined})
x = `registration`	*rule refinement* **Evaluation:** $\texttt{SAR}^{registration}_{mined}$ refines the original staff assignment rule, i.e., $\texttt{SAR}^{registration}_{mined} \equiv \texttt{SAR}^{original}_{mined}$ `OR SR` with `SR` ⟵ `(Role="secretary")` ⟹ the set of staff assignment rules can be optimized by replacing $\texttt{SAR}^{registration}_{original}$ by $\texttt{SAR}^{registration}_{mined}$
x = `examine`	*rule match* **Evaluation:** $\texttt{SAR}^{examine}_{original}$ is not adapted since its validity is supported by the mining results
x = `take blood`	*rule complement* **Evaluation:** Mined rule $\texttt{SAR}^{takeblood}_{mined}$ is completely complementary to $\texttt{SAR}^{takeblood}_{original}$ based on the underlying organizational model; $\texttt{SAR}^{takeblood}_{original}$ must not be replaced by $\texttt{SAR}^{takeblood}_{mined}$ Result may indicate that $\texttt{SAR}^{takeblood}_{original}$ is too unspecific and therefore should be reviewed.
x = `evaluate`	*rule complement* **Evaluation:** Mined rule $\texttt{SAR}^{evaluate}_{mined}$ is completely complementary to $\texttt{SAR}^{evaluate}_{original}$ based on the underlying organizational model; $\texttt{SAR}^{evaluate}_{original}$ must not be replaced by $\texttt{SAR}^{evaluate}_{mined}$ Result may indicate that $\texttt{SAR}^{takeblood}_{original}$ is too unspecific and therefore should be reviewed.
x = `special therapy`	*rule complement* **Evaluation:** Mining finds matching parts and partially complementary parts; $\texttt{SAR}^{specialtherapy}_{original}$ must not be replaced by $\texttt{SAR}^{specialtherapy}_{mined}$ $\texttt{SAR}^{specialtherapy}_{original}$ may contain unspecific parts which might be reviewed.
x = `group therapy`	*rule complement* **Evaluation:** Mining finds matching parts and partially complementary parts; $\texttt{SAR}^{grouptherapy}_{original}$ must not be replaced by $\texttt{SAR}^{grouptherapy}_{mined}$ $\texttt{SAR}^{grouptherapy}_{original}$ may contain unspecific parts which might be reviewed.
x = `write report`	*rule match* **Evaluation:** $\texttt{SAR}^{writereport}_{original}$ is not adapted since its validity is supported by the mining results
x = `lock rooms`	*rule match* **Evaluation:** $\texttt{SAR}^{lockrooms}_{original}$ is not adapted since its validity is supported by the mining results

Figure 19 shows the mining results for the first scenario where activity `examine` is executed by agent `D4` in 5% of the cases. For the best attribute `Ability="latin"` (followed by `Ability="edv"`) the mined staff assignment rule $\texttt{SAR}^{examine}_{mined\_k=1}$ totally differs from the original rule. Reason is that it describes the set of valid agents by their common properties (i.e., properties which agents with role `head of department` or `leader neurology` have in common with agent `D4`). However, this first mining result does not help in order to derive the additional substitution part. By contrast, the other mining result for $k = 3$ depicted in Figure 19 is more suitable. Here the original rule is still present but the mined rule $\texttt{SAR}^{examine}_{mined\_k=3}$ contains an additional rule

```
SR1 ⟵ ((Role="doctor") AND
  (Ability="pharmaceutic knowledge) AND (Ability="edv")
    AND (NOT (Role="head of department") AND NOT (Role="leader neurology")))
```

with:

$$\texttt{SAR}^{examine}_{mined} \equiv \texttt{SAR}^{examine}_{original} \texttt{ OR SR1}$$

The valid agent set of `SR1` contains agent `D4` and is disjoint with the valid agent set of $\texttt{SAR}^{examine}_{original}$. Consequently, `SR1` complements the valid agent set of $\texttt{SAR}^{examine}_{original}$ by substitution agent `D4`. Therefore `SR1` can be used to specify a substitution rule for $\texttt{SAR}^{examine}_{original}$.

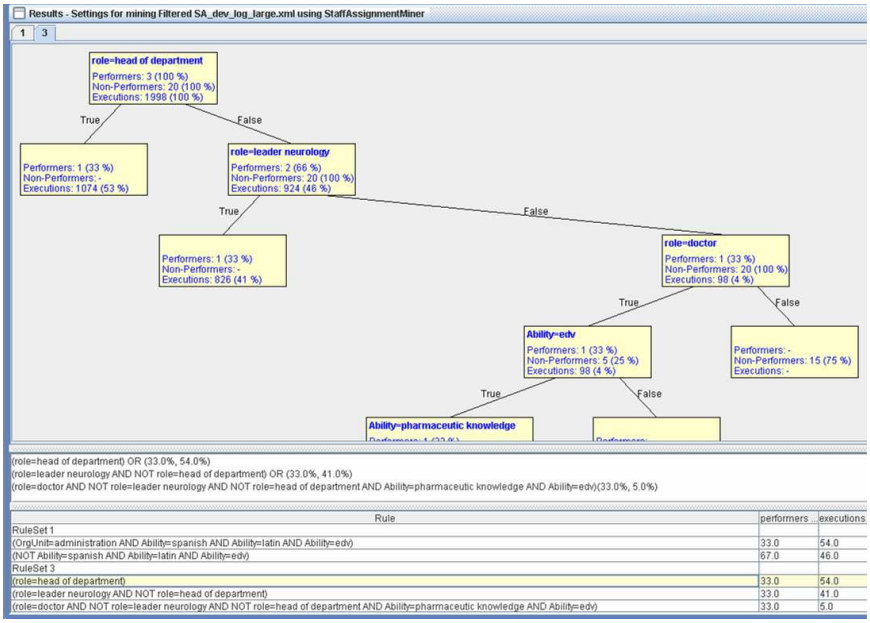


Fig. 19. Mining result for activity `examine` with 5% deviation.

After mining the simulation logs for the second scenario (activity `write report`) we observe that the mined staff assignment rule for the best attribute $\texttt{SAR}^{writereport}_{mined\_k=1}$ contains additional rule

`SR2` ⟵ `(NOT (Role="doctor") AND (Ability="pharmaceutic knowledge") AND (Ability="edv"))`

(cf. Figure 20) with $\texttt{SAR}^{writereport}_{mined\_k=1} \equiv \texttt{SAR}^{writereport}_{original}$ `OR SR2`.
For the valid agent set of `SR2`, VAS`(OM, SR2)` = {`SS`} holds. `SS`, in turn, is the only agent having role `staff nurse` over organizational model OM (as specified for exceptional execution of `write report` in our example). Therefore rule `SR2` can be suggested as substitution rule for original rule $\texttt{SAR}^{writereport}_{original}$.

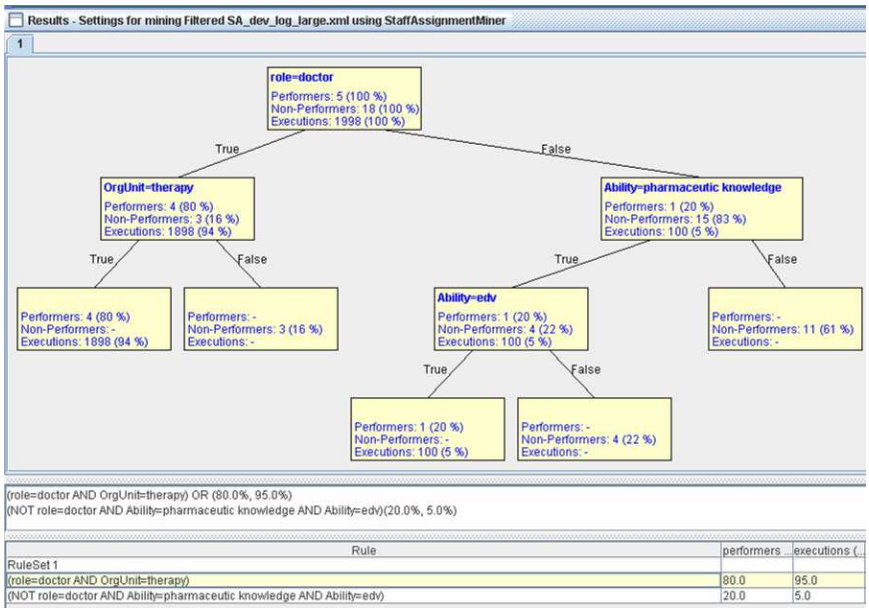


Fig. 20. Mining result for activity `write report` with 5% deviation

When comparing substitution rule `SR2` to the associated original substitution rule `R2` we observe that `SR2` is complementary to `R2` (cf. Section 4.3). Since the existence and specification of `R2` are just assumptions of the simulation scenario, it makes sense to suggest complementary substitution rules to users as well (contrary to the evaluation suggestions presented in Section 4.3 where complementary rules must not replace the original ones). To prove their quality an a-posteriori analysis based on LTL-checking can be applied (i.e., we can check whether the agents have followed the substitution rules in exceptional cases or not). A still open question is how to specify the substitution rules within a staff assignment rule by emphasizing their exceptional character. If the substitution

rule is simply connected to the original staff assignment rule in a disjunctive way, the resulting rule may be too unspecific afterwards. Therefore the definition of staff assignment rules should be extended with respect to capture "standard" rules as well as substitution parts. However, this topic is outside the scope of this paper.

5 Related Work

In this section we discuss related work on process mining (cf. Section 5.1) followed by approaches in the field of organizational aspects in PAISs (cf. Section 5.2).

5.1 Process Mining

Since the early nineties, workflow technology has matured [29] and several textbooks have been published, e.g., [4, 25]. A wide variety of languages has been proposed typically focusing on the control-flow and ranging from Petri nets [4, 25] to BPEL [14] (cf. the work on workflow patterns [5] where different languages are compared). Many authors point out that flexibility is an important issue [12, 46, 48]. As indicated in [26] coupling flexibility and mining promises interesting perspectives towards and intelligent user support (e.g., for discovery and definition of changes). Therefore, we have implemented a link between the change logs in ADEPT [46, 48] and ProM [22].

This paper builds on earlier work on process mining. Classically, the focus of process mining has been on control-flow discovery. Many algorithms have been proposed, e.g., [11, 13, 20, 35]. Few people have been focusing on the mining of the organizational perspective, except the work done in [8, 7]. Here social networks are constructed on the basis of MXML logs. Similarly, little work has been done on the data perspective. The decision miner is a notable exception [51], it can discover the rules for decision points. When it comes to check a log with respect to some a-priori model, the conformance checker [50] and the LTL checker [2] are two examples. It is impossible to give a complete overview of process mining here. Therefore, we refer to a special issue of Computers in Industry on process mining [10] and a survey paper [9].

Process mining can also be seen in the broader context of Business (Process) Intelligence (BPI) and Business Activity Monitoring (BAM). In [31, 32, 52] a BPI toolset on top of HP's Process Manager is described. The BPI toolset includes a so-called "BPI Process Mining Engine". In [42] Zur Muehlen describes the PISA tool which can be used to extract performance metrics from workflow logs. Similar diagnostics are provided by the ARIS Process Performance Manager (PPM) [36]. The latter tool is commercially available and a customized version of PPM is the Staffware Process Monitor (SPM) [54] which is tailored towards mining Staffware logs. Interestingly, process mining is also getting attention in the domain of web services. In [30], Dustdar et al. discuss the concept of web services mining and envision various levels (web service operations, interactions, and workflows) and approaches. In [43] a tool named the Web Service Navigator

is presented to visualize the execution of web services based on SOAP messages. In [3] we applied the conformance checking techniques described in [50] to web services logs.

5.2 Organizational Aspects

The provision of adequate access control mechanism is indispensable for any cooperative information system. In the literature many approaches deal with corresponding issues (e.g., [15, 58, 61]). Most of them use *Role-Based Access Control (RBAC)* models for defining and managing user privileges [15, 27, 28] (e.g., for ensuring the controlled access to business documents when using *document management* technology or for resolving the set of agents that qualify for a certain task defined by a staff assignment rule in a *workflow management system* [17, 16, 56, 61, 58]). In addition, dynamic constraints (e.g., separation of duties) [16, 56] and the evolution of organizational structures [37, 47, 55] have been considered. However, the mentioned approaches neither show how to obtain organizational structures and related staff assignment rules automatically nor how to evaluate their quality during their life-cycle.

The theoretical background for staff assignment mining has been presented in [39] and is extended in several directions within this paper. First of all, the basic concepts have been implemented as a plug-in in the context of the ProM framework and tested based on an extensive simulation study. From this, we derived and formalized evaluation criteria for the mined staff assignment rules based on which the original staff assignment rules can be optimized. Furthermore it was shown how to detect deviations from the original staff assignment rules and formalize them by so-called substitution rules.

6 Summary and Outlook

User support in defining and optimizing all aspects of business processes is key ingredient for the practical applicability of future PAISs. Together with our previous work on process mining, business process evolution, and dynamic process change [9, 46, 49] the presented concepts contribute to a powerful platform enabling the user-friendly realization of flexible and adaptive information systems.

In this paper we have developed a framework for the complete life-cycle support for staff assignment rules. Our approach for staff assignment mining is based on audit trail data, knowledge about organizational structures, and a decision tree learning method [39]. It has been implemented as a plug-in in the context of the ProM framework. Based on the simulation of an example clinical process using CPN tools and ProMimport, the staff assignment mining approach has been evaluated with respect to the quality of the pre-defined staff assignment rules. We defined and investigated the possible relations between original and mined staff assignment rules. In addition, it has been shown how these relations can be used to derive optimization strategies afterwards. Furthermore, the simulation example has been extended by capturing exceptional agent behavior

as well. After analyzing the corresponding data using staff assignment mining we have shown how substitution rules can be specified. Using this approach all phases of the staff assignment life-cycle can be supported in an adequate way.

There are many other challenging issues that can be linked to the definition and evolution of staff assignment rules. First of all, we intend to analyze exceptional agent behavior in connection with conditions imposed on the process. Assume that in the treatment process (cf. Figure 4), for example, activity `special therapy` is executed by the staff nurse but only if the blood value has exceeded 100, i.e., only for exceptional cases a specific staff assignment takes place. Using this information we could come to even more precise definitions of staff assignment rules. Furthermore, the discovery and analysis of *dependent* staff assignment rules (i.e., rules which are dynamically resolved during runtime) is interesting from a practical point of view. An example for a dependent staff assignment rule could be $\texttt{SAR}_{dep}$`(write report)` ⟵ `Role=sameRole(evaluate)` where `sameRole` implies that activity `write report` is to be executed by an agent having the same role as the agent having executed activity `evaluate`.

Altogether, a seamless import of staff assignment optimizations becomes possible when integrating staff assignment mining with our work on the controlled evolution of organizational models [47]. In [47] we have shown how staff assignment rules are affected after changes of the organizational model. Currently we are working on the question how the set of valid agents is influenced after changing staff assignment rules. This could be also used for directly applying staff assignment changes based on the mining and evaluation approach presented in this paper within the PAIS.

Acknowledgment: The `StaffAssignmentMiner` plug-in of the ProM framework was implemented within a student project at the department DBIS at Ulm University. We thank Martin Bader for the implementation work and Linh Thao Ly for the co-supervision of this project.

References

1. W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270(1-2):125–203, 2002.
2. W.M.P. van der Aalst, H.T. de Beer, and B.F. van Dongen. Process Mining and Verification of Properties: An Approach based on Temporal Logic. In R. Meersman and Z. Tari et al., editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, volume 3760 of *Lecture Notes in Computer Science*, pages 130–147. Springer-Verlag, Berlin, 2005.
3. W.M.P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H.M.W. Verbeek. Choreography Conformance Checking: An Approach based on BPEL and Petri Nets (extended version). BPM Center Report BPM-05-25, BPMcenter.org, 2005.
4. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.

5. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
6. W.M.P. van der Aalst, A.K. Alves de Medeiros, and A.J.M.M. Weijters. Genetic Process Mining. In G. Ciardo and P. Darondeau, editors, *Applications and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 48–69. Springer-Verlag, Berlin, 2005.
7. W.M.P. van der Aalst, H.A. Reijers, and M. Song. Discovering Social Networks from Event Logs. *Computer Supported Cooperative work*, 14(6):549–593, 2005.
8. W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering Interaction Patterns in Business Processes. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 244–260. Springer-Verlag, Berlin, 2004.
9. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
10. W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
11. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
12. W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.
13. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
14. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.1. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.
15. E. Bertino. Data security. *Data & Knowledge Engineering*, 25(1–2):199–216, March 1998.
16. E. Bertino, E. Ferrari, and V. Alturi. The specification and enforcement of authorization constraints in wfms. *ACM Transactions on Information and System Security*, 2(1):65–104, 1999.
17. R.A. Botha and J.H.P. Eloff. A framework for access control in workflow systems. *Information Management and Computer Security.*, 9(3):126–133, 2001.
18. L. Breslow and D. Aha. Simplifying decision trees: a survey. *Knowledge Engineering Review*, 12(1):1–40, 1997.
19. S. E. Clausen. *Applied Correspondence Analysis: An Introduction.* Sage Publications, 1998.
20. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
21. CPN Group, University of Aarhus, Denmark. CPN Tools Home Page. http://wiki.daimi.au.dk/cpntools/, 2006.

22. B. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.
23. B.F. van Dongen and W.M.P. van der Aalst. Multi-Phase Process Mining: Building Instance Graphs. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T.W. Ling, editors, *International Conference on Conceptual Modeling (ER 2004)*, volume 3288 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, Berlin, 2004.
24. B.F. van Dongen and W.M.P. van der Aalst. A Meta Model for Process Mining Data. In J. Casto and E. Teniente, editors, *Proceedings of the CAiSE'05 Workshops (EMOI-INTEROP Workshop)*, volume 2, pages 309–320. FEUP, Porto, Portugal, 2005.
25. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
26. S. Dustdar, T. Hoffmann, and W.M.P. van der Aalst. Mining of ad-hoc business processes with TeamLog. *Data and Knowledge Engineering*, 55(2):129–158, 2005.
27. D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli. *Role-Based Access Control*. Artech House, 2003.
28. D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.
29. D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.
30. R. Gombotz and S. Dustdar. On Web Services Mining. In M. Castellanos and T. Weijters, editors, *First International Workshop on Business Process Intelligence (BPI'05)*, pages 58–70, Nancy, France, September 2005.
31. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan. Business Process Intelligence. *Computers in Industry*, 53(3):321–343, 2004.
32. D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. In P. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. Snodgrass, editors, *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 159–168. Morgan Kaufmann, 2001.
33. C.W. Guenther and W.M.P. van der Aalst. Modeling the Case Handling Principles with Colored Petri Nets. In K. Jensen, editor, *Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2005)*, volume 576 of *DAIMI*, pages 211–230, Aarhus, Denmark, October 2005. University of Aarhus.
34. D. Hand, H. Mannila, and P. Smyth. *Priciples of Data Mining*. MIT Press, 2001.
35. J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.
36. IDS Scheer. ARIS Process Performance Manager (ARIS PPM): Measure, Analyze and Optimize Your Business Process Performance (whitepaper). IDS Scheer, Saarbruecken, Gemany, http://www.ids-scheer.com, 2002.
37. J. Klarmann. A comprehensive support for changes in organizational models of workflow management systems. In *Proc. Int'l Conf. on Information Systems Modeling (ISM'01)*, Hradec nad Moravici, Czech Republic, May 2001.

38. K. Kochut, J. Arnold, A. Sheth, J. Miller, E. Kraemer, B. Arpinar, and J. Cardoso. IntelliGEN: A distributed workflow system for discovering protein-protein interactions. *Distributed and Parallel Databases*, 13(1):43–72, 2003.
39. T. Ly, S. Rinderle, P. Dadam, and M. Reichert. Mining staff assignment rules from event-based data. In M. Castellanos and T. Weijters, editors, *First International Workshop on Business Process Intelligence (BPI'05)*, pages 177–190, Nancy, France, September 2005.
40. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1991.
41. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
42. M. zur Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
43. W. De Pauw, M. Lei, E. Pring, L. Villard, M. Arnold, and J.F. Morar. Web Services Navigator: Visualizing the Execution of Web Services. *IBM Systems Journal*, 44(4):821–845, 2005.
44. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
45. J. Quinlan. Learning decision tree classifiers. *ACM Computing Surveys*, 28(1):71–72, 1996.
46. M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
47. S. Rinderle and M. Reichert. On the controlled evolution of access rules in cooperative information systems. In R. Meersman and Z. Tari et al., editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, volume 3760 of *LNCS*, pages 238–255. Springer, 2005.
48. S. Rinderle, M. Reichert, and P. Dadam. Correctness Criteria For Dynamic Changes in Workflow Systems: A Survey. *Data and Knowledge Engineering*, 50(1):9–34, 2004.
49. S. Rinderle, M. Reichert, and P. Dadam. Flexible support of team processes by adaptive workflow systems. *Distributed and Parallel Databases*, 16(1):91–116, 2004.
50. A. Rozinat and W.M.P. van der Aalst. Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In C. Bussler et al., editor, *BPM 2005 Workshops (Workshop on Business Process Intelligence)*, volume 3812 of *Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, Berlin, 2006.
51. A. Rozinat and W.M.P. van der Aalst. Decision Mining in Business Processes. BPM Center Report BPM-06-10, BPMcenter.org, 2006.
52. M. Sayal, F. Casati, U. Dayal, and M.C. Shan. Business Process Cockpit. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 880–883. Morgan Kaufmann, 2002.
53. J. Scott. *Social Network Analysis*. Sage, Newbury Park CA, 1992.
54. TIBCO. TIBCO Staffware Process Monitor (SPM). http://www.tibco.com, 2005.
55. W.M.P. van der Aalst and S. Jablonski. Dealing with workflow change: Identification of issues an solutions. *Int'l Journal of Comp. Systems, Science and Engineering*, 15(5):267–276, 2000.

56. J. Wainer, P. Barthelmess, and A. Kumar. W–RBAC – a workflow security model incorporating controlled overriding of constraints. *International Journal of Collaborative Information Systems*, 12(4):455–485, 2003.
57. S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, 1994.
58. B. Weber, M. Reichert, W. Wild, and S. Rinderle. Balancing flexibility and security in adaptive process management systems. In R. Meersman and Z. Tari et al., editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, volume 3760 of *LNCS*. Springer, 2005.
59. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
60. M. Weske. Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In *Proc. Hawaii International Conference on System Sciences (HICSS-34)*, 2001.
61. M. zur Muehlen. Resource modeling in workflow applications. In *Proc. of the 1999 Workflow Management Conference (Muenster)*, pages 137–153, 1999.