Mining Staff Assignment Rules from Event-Based Data

Linh Thao Ly[1], Stefanie Rinderle[1], Peter Dadam[1], and Manfred Reichert[2]

[1] Dept. DBIS, University of Ulm, Germany
{thao.ly, rinderle, dadam}@informatik.uni-ulm.de
[2] IS Group, University of Twente, The Netherlands
m.u.reichert@cs.utwente.nl

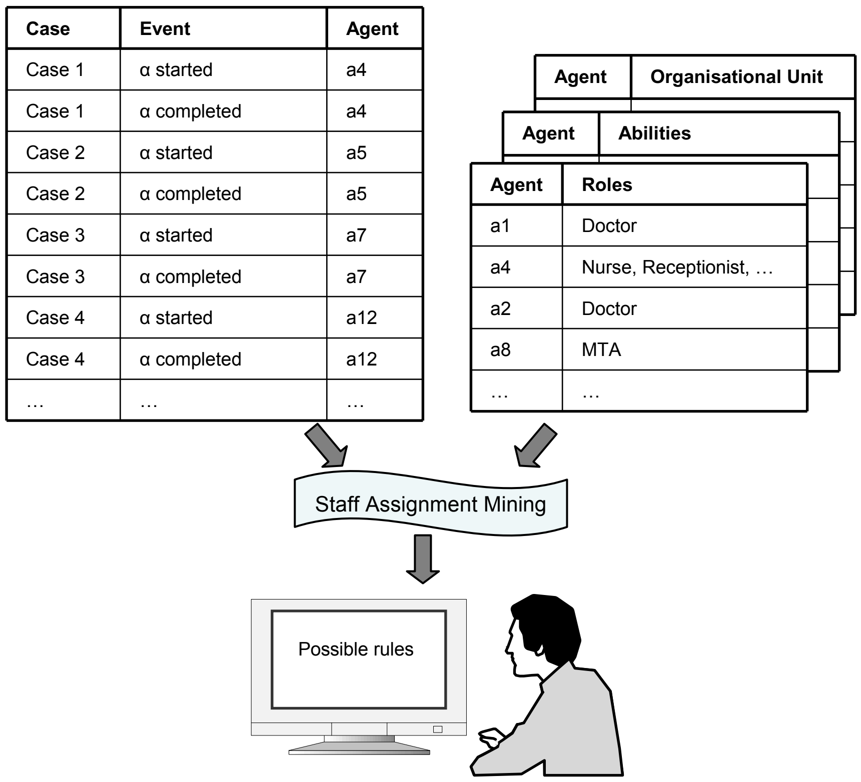
Abstract. Process mining offers methods and techniques for capturing process behaviour from log data of past process executions. Although many promising approaches on mining the control flow have been published, no attempt has been made to mine the staff assignment situation of business processes. In this paper, we introduce the problem of mining staff assignment rules using history data and organisational information (e.g., an organisational model) as input. We show that this task can be considered an inductive learning problem and adapt a decision tree learning approach to derive staff assignment rules. In contrast to rules acquired by traditional techniques (e.g., questionnaires) the thus derived rules are objective and show the staff assignment situation at hand. Therefore, they can help to better understand the process. Moreover, the rules can be used as input for further analysis, e.g., workload balance analysis or delta analysis. This paper presents the current state of our work and points out some challenges for future research.

1 Introduction

While great effort has been spent on researching the control flow aspect of business processes, organisational aspects of processes have often been neglected. In particular, the link between the process and the organisational elements is less understood [1]. However, in order to fully understand a business process, it is also necessary to know by whom the activities of the process are performed. This especially applies when *Workflow Management Systems* (WfMS) should be employed in order to support the process execution. In WfMSs rules which are based on organisational concepts, e.g., roles, are often used to assign work items to staff members (staff assignment rules). Staff assignment rules define, to a certain extent, the profile of agents capable of or eligible for performing an activity. For example, for performing the activity “create bills” agents have to possess the role “book-keeper” and additionally need to have “computer skills”. Properties not referred to in the staff assignment rule have don’t-care semantics. An agent might have those properties or not.

The traditional way of acquiring staff assignment rules (or process knowledge in general) is by means of questionnaires or interviews. However, these techniques are very cost-intense and error-prone. Furthermore, the results acquired by applying traditional techniques are rather subjective and need not necessarily reflect the staff assignment situation at hand. Therefore, it makes sense to support the process engineer in acquiring staff assignment rules by providing objective rules as a complement to the results acquired by traditional methods.

In this paper, we introduce the task of deriving staff assignment rules from log data of past process executions and organisational information. We denote this as *staff assignment mining*.



Case	Event	Agent
Case 1	α started	a4
Case 1	α completed	a4
Case 2	α started	a5
Case 2	α completed	a5
Case 3	α started	a7
Case 3	α completed	a7
Case 4	α started	a12
Case 4	α completed	a12
…	…	…

Agent	Roles
a1	Doctor
a4	Nurse, Receptionist, …
a2	Doctor
a8	MTA
…	…

Fig. 1. The process engineer is supported by proposing a set of possible staff assignment rules for a given activity

Workflow Management Systems but also other process-oriented systems, e.g., *Enterprise Ressource Planning Systems* (ERP) like SAP, log all events which occur while a process instance (a *case*) is executed. The log data, also called audit trail or history data, typically contain information about the start and the end of an activity but also about the agent who performed the activity. By combining this log data with organisational information (for instance, from an organisational model), e.g., the roles that staff members have, objective staff assignment rules can be derived (cf. Fig. 1).

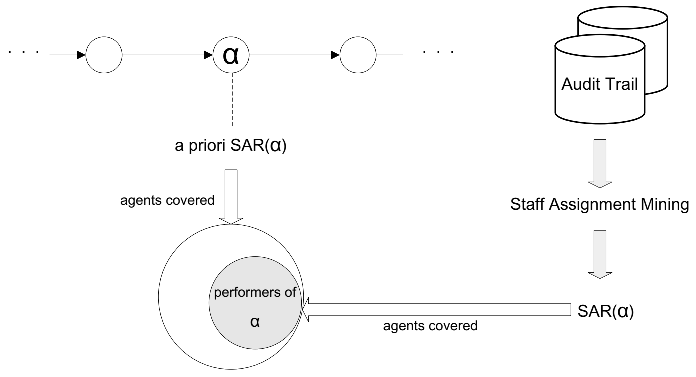


Fig. 2. Using staff assignment mining for delta-analysis

Fig. 1 shows some log data of the execution of an example activity α in different cases. Abstracting from concrete events we will refer to a performer of an activity as an agent who started and completed an instance of the activity.

By using organisational concepts (e.g., roles) for separating the performers of the activity from the non-performers[1] meaningful staff assignment rules can be derived. If we for instance find out that all performers of the example activity α, e.g., a4, a5 etc., have the role "doctor" while all non-performers do not, it is likely that the role "doctor" is a key property for performing α. Thus, the staff assignment rule for α could demand that all agents need to possess the role "doctor".

Our objective is to derive staff assignment rules for a given activity such that:

- the rules are consistent to the audit trail data
- the rules identify the actual set of performers of the activity
- the rules are general such that they cover the essential profile of the performers

The derived staff assignment rules reveal the actual profile of the performers and thus, reveal the staff assignment situation at hand. Therefore, they can be used as input for further analysis, particularly delta-analysis (cf. Fig. 2). If an a priori staff assignment rule of the example activity α (SAR(α)) in Fig. 2 is, for instance, more general than the rule derived, this indicates that only a subset of the agents identified by the a priori rule really performed instances of α. This might indicate that the work item is delivered to work queues of staff members who never performed the activity. This, of course, can be intended. However, it might also indicate that the staff assignment situation has changed and that the a priori rule is obsolete. Besides a better understanding of the process behaviour knowing the actual staff assignment situation at hand also allows for incrementally defining staff assignment rules.

This paper is organised as follows. After addressing related work in the process mining context in Sect. 2 we will refer to the problem of learning staff

[1] Agents who did not perform the activity.

assignment rules. For this purpose an organisational meta-model and an appropriate representation of staff assignment rules are introduced in Sect. 3. Then, the problem of learning staff assignment rules is addressed in Sect. 4. Finally, Sect. 5 concludes the paper.

2 The Process Mining Context

Besides staff assignment rules, many information can be derived from the audit trail data of process executions. *Process mining* deals with developing methods and techniques to capture the process behaviour from audit trail data. In particular, a structured process description can be derived using *process mining* techniques. The derived process model can be used as input for further analysis. For instance, delta-analysis can be applied to detect discrepancies between the a priori and the derived process model.

Many papers on *process mining* have been published recently (e.g., [6, 7, 10, 12, 14, 20, 11]). Most of them focus on mining the control flow (*control flow mining*). Only few papers consider organisational aspects. For a survey on *process mining* approaches the reader is referred to [3, 2].

Approaches integrating organisational aspects can be divided into two categories. The first category concentrates on relations between agents involved in the process [4, 5]. The second category focuses on the relations between a process and organisational concepts. Our approach presented in this paper falls in the second category.

In [4, 5] van der Aalst and Song introduce an approach for mining social networks from log data. The authors define four categories of metrics expressing potential relationships between agents (e.g., metrics based on joint activities). Using these metrics sociograms are derived which can be further used for *social network analysis*. This work can be considered an important contribution to *enterprise social networks analysis*.

The authors also mentioned the possibility of "guessing" organisational structures, in particular, guessing roles of agents. Agents performing the same activities are assigned the same roles. However, in [4, 5], van der Aalst and Song do not consider the use of additional organisational information in this context. In addition, it seems that this was just a suggestion since no further work on this aspect has been published. At our best knowledge, no other work on mining the relations between the process and the organisation is available to date.

Staff assignment mining is a novel facet of *process mining* and can be smoothly integrated in the mining process. We consider our approach a complement to current process mining efforts.

3 The Organisational Meta-model

As a starting point, we use a simple but yet powerful organisational meta-model to describe organisational concepts (cf. Fig. 3). However, the approach presented in this paper is not restricted the meta-model and the organisational concepts presented here. In fact, our approach can be directly applied for other organisational meta-models and further organisational concepts as well. In order to present our approach, however, a meta-model is needed to specify the organisational concepts used. The meta-model uses the following organisational

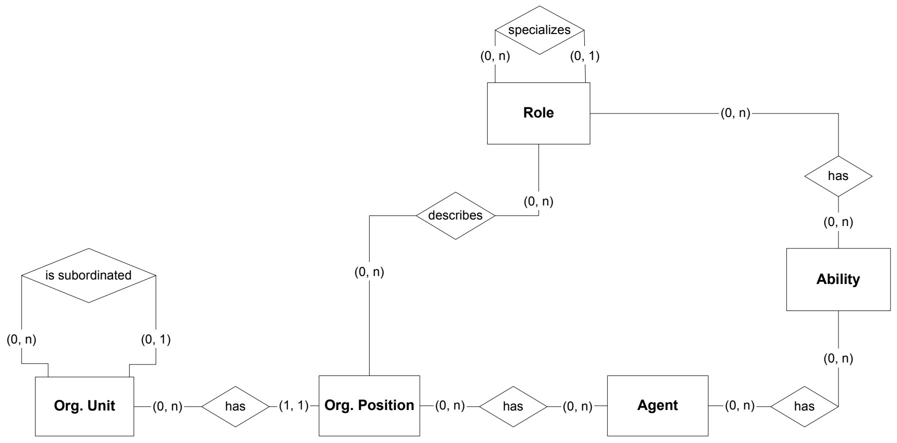


Fig. 3. The organisational meta-model used

concepts: agents, organisational units, roles, abilities, and organisational positions. Since the latter one is considered to be assigned to only one agent (except for time-sharing aspects) we neither consider agent objects nor organisational positions when deriving rules, since those rules would not represent a general profile. Abilities can be assigned to agents directly or indirectly via roles. Being assigned a certain role an agent also has all abilities, i.e. capabilities and privileges, associated with that role. For example, the role “receptionist” has the ability “computer skills”. Organisational positions, e.g., “1st book-keeper”, can be interpreted as an instantiation of a set of roles. Due to space limitations we cannot go into detail on the meta-model. For further information the reader is referred to [18]. Table 1 shows an example of an organisation model based on this meta-model. We will refer to this example later.

Based on the organisational meta-model, staff assignment rules (abbr.: SAR) can refer to organisational entities in a manner similar to disjunctive normal forms (DNF) in order to define the profile of the performers. A SAR of the example activity α (SAR(α)) is given below, where a certain role (ability) is specified by `R` (`A`). This rule would identify the agents a4, a5, a7, a11 and a12 from Tab. 1.

```
SAR(α):
    (R = 'receptionist' AND A = 'english')
    OR
    (R = 'receptionist' AND A = 'french')
```

Table 1. Example of an organisational model. The first part lists agents and respective organisational entity, position and roles (MTA stands for medical-technical assistant). The second part lists agents and respective abilities. Abilities directly assigned to the agent are marked with an asterisk.

Agent	Org. unit	Org. position	Roles
A1	Clinical Centre	1st Doctor	Doctor
A2	Clinical Centre	2nd Doctor	Doctor
A3	Clinical Centre	3rd Doctor	Doctor
A4	Clinical Centre	1st Nurse	Nurse, Receptionist, Book-keeper
A5	Clinical Centre	2nd Nurse	Nurse, Receptionist, Book-keeper
A6	Clinical Centre	3rd Nurse	Nurse, Receptionist, Book-keeper
A7	Clinical Centre	4th Nurse	Nurse, Receptionist, Book-keeper
A8	Clinical Centre	1st MTA	MTA
A9	Clinical Centre	2nd MTA	MTA
A10	Clinical Centre	3rd MTA	MTA
A11	Clinical Centre	1st Secretary	Secretary, Receptionist, Book-keeper
A12	Clinical Centre	2nd. Secretary	Secretary, Receptionist, Book-keeper

Agent	Abilities
A1	Computer skills*, Take blood sample, Issue prescription, English*
A2	Computer skills*, Take blood sample, Issue prescription
A3	Computer skills*, Take blood sample, Issue prescription, English*
A4	Computer skills, Take blood sample, English*, French*
A5	Computer skills, Take blood sample, English*
A6	Computer skills, Take blood sample
A7	Computer skills, Take blood sample, French*
A8	English*
A9	English*
A10	
A11	Computer skills, French*
A12	Computer skills, English*

Note that it is also possible to use negative qualifications by using NOT, in the sense of demanding that an agent must not have certain properties. If an agent is not related to an organisational entity, then he is considered to have negative qualifications concerning these entities. Though organisational entities may have many attributes (for example, the ability “english” may have the attribute “level” with values ranging from “beginner” to “expert”) we abstract from attributes other than the name of the entity.

Rules in the form described above can be used to define the profile of appropriate performers of a workflow activity but also to define access rules or constraints for any kind of information system and objects (e.g., to control the access to electronic documents) [19, 22].

4 Learning Staff Assignment Rules

In this section, we present our approach for deriving meaningful staff assignment rules using audit trail data and organisational information based on the meta-model described before.

4.1 Decision Tree Learning

Since staff assignment rules are supposed to identify the performers of a given activity, the question is to determine combinations of organisational properties that distinguish performers from non-performers. Thus, the problem of deriving staff assignment rules can be interpreted as an inductive learning task, particularly learning from positive and negative examples. 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 1 (Positive/Negative Examples). *Let A be a set of agents, and let X be the total set of activities. Then performer(x,a) 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:*

$$performer : X \times A \rightarrow \{\texttt{True}, \texttt{False}\}$$

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

An 'example' is a triple $(x, a, performer(x,a))$. We further distinguish between positive examples, i.e., $(x, a, \texttt{True})$, and negative examples, i.e., $(x, a, \texttt{False})$. 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.

Table 2 shows a set of examples referring to the agents from our organisational model depicted in Tab. 1 and our example activity α (cf. Fig. 1). Since we refer to α, our running example throughout this paper, the activity information is omitted in Tab. 2.

Based on the examples the objective is to derive rules which approximates the classification function *performer*. This problem belongs to *supervised learning* [13] since we have predefined classes (performers and non-performers).

Various learning methods can be applied to solve this learning problem. We have chosen to adapt *decision tree learning* [9]. *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 facilitates 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

Table 2. A set of examples. The agents a1, a2, a6, a8, a9, and a10 did not perform α while the agents a4, a5, a7, a11, and a12 did.

Agent a	performer(α,a)
a1	`False`
a2	`False`
a3	`False`
a4	`True`
a5	`True`
a6	`False`
a7	`True`
a8	`False`
a9	`False`
a10	`False`
a11	`True`
a12	`True`

handling noise data and continuous attribute values. Continuous attribute values do not occur with our preliminaries. However, this will be an important feature when we will extend our approach to consider attributes of organisational entities as well.

Staff assignment rules can be derived by growing decision trees (cf. Fig. 4). Starting at the root node, an organisational entity is chosen as testing attribute in order to separate the positive from the negative examples. In Fig. 4 `R = 'receptionist'` was chosen as the first attribute. (Which attributes are chosen and in which order is discussed in the following.) Depending on whether they are related to an organisational 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 organisational entity or not. This procedure is repeated recursively for the child nodes until only examples from one class, indicated by the '+' and the '−' set in Fig. 4, are left, or there are no attributes left. The '+' set represents the class of performers while the '−' set represents the class of non-performers.

All entities of an organisational model can be used as testing attributes. For example, the set of possible attributes shown below can be derived from the organisational model described in Tab. 1. A certain organisational unit is specified by `OU`.

```
{OU = 'clinical centre', R = 'nurse', R = 'doctor',
  R = 'receptionist', ...}
```

From a decision tree if-then-rules or rules in 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.

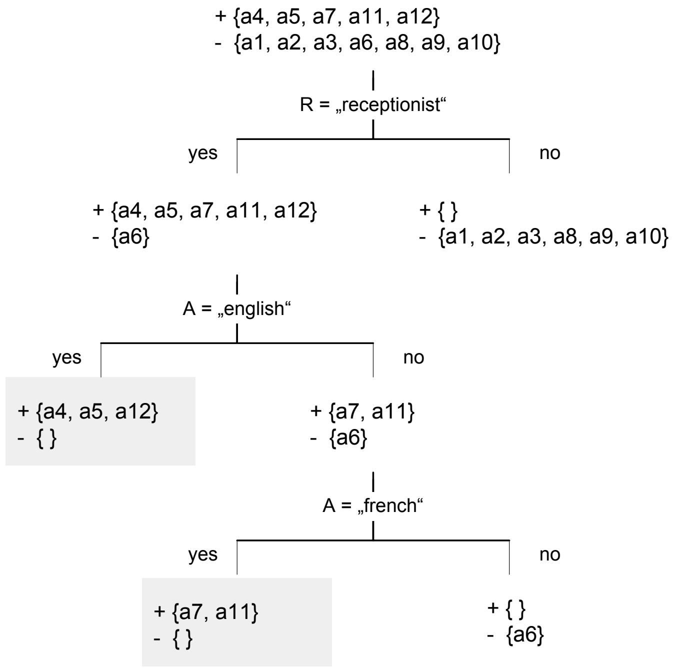


Fig. 4. A decision tree for the example set for activity α from Tab. 2. From this decision tree the rule `SAR(`α`): (R = 'receptionist' AND A = 'english') OR (R = 'receptionist' AND A = 'french')` can be derived.

However, our objective is to mine general profiles of performers with as less conjunction elements as possible. Finding decision trees representing minimal rules is of NP-hard complexity [16]. Therefore, a gready search strategy using the metrics *information gain* [17, 16] for guiding the search, i.e. choosing an attribute, is applied. The *information gain* metrics is based on entropy calculations. The formulas for calculating the entropy and *information gain* are given below. S denotes 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 each separation step the attribute with the best *information gain* value is chosen. Thus, the decision tree algorithm tries to achieve the best split of the remaining example set in every step.

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

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

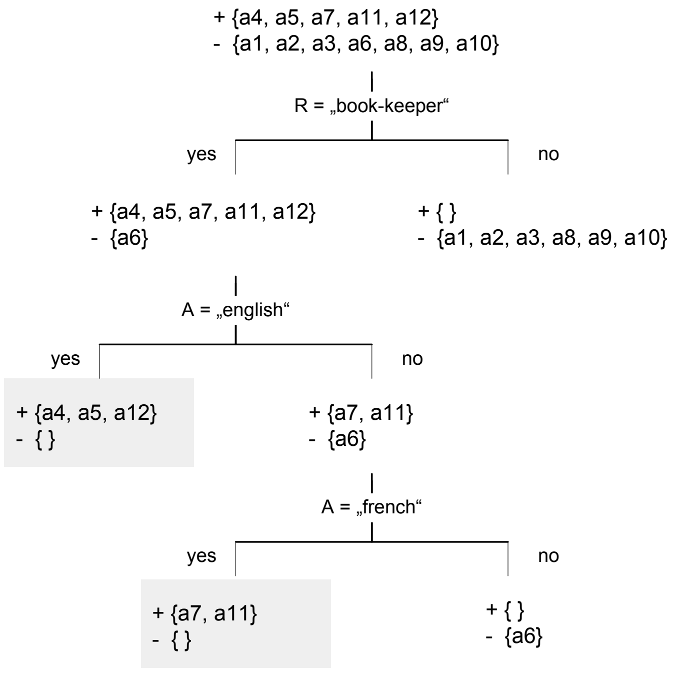


Fig. 5. An alternative decision tree for the example data. From this decision tree the rule `SAR(`α`): R = ('book-keeper' AND A = 'english') OR (R = 'book-keeper' AND A = 'french')` can be derived.

The decision tree in Fig. 4 was generated using *information gain.* For further information on decision trees and metrics please refer to [17, 16, 15].

Generally, more than one decision tree can often be derived which fit the input data. This also applies to our example set from Tab. 2. Therefore, it is important to offer a set of potential rules to the process engineer. The process engineer can then, after an evaluation process, decide which of the rules are useful. In order to extract more than one rule, backtracking is needed. Again, *information gain* can be used for choosing the suitable attributes. Instead of using only the best separating attribute, the k-best attributes can be used, whereas k is a configurable parameter. Figure 5 shows an alternative decision tree using the second best testing attribute (R= 'book-keeper') at root level. Note that besides the trees shown here even more decision trees can be derived from our example set by choosing another k and/or allowing backtracking on other than the root level.

4.2 Advanced Issues

Attributes, i.e. organisational entities, are, with regard to the given organisational meta-model, not necessarily independent. In fact, attributes can be related to each other in different ways. Organisational units can consist of other units. Thus, any agent in the sub-unit also belongs to the superordinate unit. Furthermore, roles can be in a specialisation/generalisation relationship to other roles. For example, the role “nurse” can have the role “lead nurse” as specialisation, which inherits all privileges and abilities of the role “nurse”. Thus, having the role “lead nurse” directly implies having the role “nurse”.

Another case of dependent attributes occurs with roles and abilities. Roles can imply abilities but not vice versa. Since every organisational entity is represented by an attribute, those dependencies need not be handled separately. However, when an attribute is selected for separation, all attributes implied by it need not be used as testing attributes because they would not achieve a further separation. Therefore, these attributes can be excluded from the set of remaining attributes for this path. This helps reducing the amount of attributes to test.

Moreover, it will typically be possible to confine the set of relevant organisational entities as well as relevant examples in advance. On the one hand, it is often possible to exclude certain organisational entities. For the activity “create bills’, for instance, ’ the ability “take a blood sample” is fairly uninteresting. Therefore, the ability “take a blood sample” can be excluded from the set of testing attributes since staff assignment rules based on this attribute would not make sense anyway.

On the other hand, a basic set of qualifications necessary for performing an activity is often already known in advance. For example, for activity “examine the patient” the role “doctor” is required. Agents, who do not have the required qualifications, can be excluded from the example set. This helps reducing the amount of examples. The task of excluding attributes in advance or selecting qualifications, which agents need to possess, should be performed by the process engineer.

4.3 Dealing with Noise

“Perfect” process executions and perfect log data as in the previous examples, however, cannot be taken for granted. Hence, we also have to deal with exceptional cases and noise data. Exceptional cases are considered to be cases, where agents perform the activity although they are not eligible to do so , e.g., as a replacement. Replacement performers do not necessarily have the profile of regular performers. Therefore, the decision tree might not reveal the profile of regular performers.

Noise data occur when, for instance, a wrong agent is logged as the performer of the activity. In order to account for those cases, threshold values are introduced. Concerning performers, the frequency of their occurrence in the example set can be used as an indication of whether they are regular performers or not. If agent a4, for instance, executed activity α twice while all other agents who executed α did so a lot more often, this indicates that agent a4 is not a regular perfomer of activity α. Thus, performers executing α less often than a given threshold value can be removed from the example set in advance. Threshold values can also be used for post-pruning the decision tree. For example, nodes where the proportion of positive examples is less than a threshold value can be transformed into a leaf-node.

Since it is our objective to identify the actual performer set, pruning the tree based on negative examples should mainly be used in order to account for minor errors of the organisational model, e.g., non-performer agents were assigned spurious properties making them more difficult to separate from the positive examples. In a post-pruning operation nodes where the amount of negative examples is less than a threshold value can be transformed into a leaf-node. For further information on pruning decision trees the interested reader is referred to [17, 9].

In contrast to *control flow mining* we use organisational information (organisational model) as input data, in addition to the audit trail data. Thus, the quality of the derived rules highly depends on the quality of the organisational model. As aforementioned, minor errors in the organisational model can be compensated using threshold values. Nevertheless, the organisational model needs to be complete, in the sense of that all relevant organisational entities are modelled. Mistakes or incompleteness of the organisational model may lead to less meaningful rules. However, the derived rules at least reveal the actual situation.

5 Conclusion and Outlook

In this paper, we concentrated on a new aspect of *process mining*: mining staff assignment rules. We have shown that the problem of deriving staff assignment rules using information from audit trail data and organisational information (e.g., an organisational model) as input can be interpreted as an inductive learning problem. Therefore, machine learning techniques can be adapted in order to solve the problem. In particular, we have used decision tree learning to derive meaningful staff assignment rules. Thus, it is possible to provide staff assignment information about activities enabling a better understanding of the underlying process.

However, enhancements and alternative learning methods have to be considered. Instead of using *information gain* as the metrics for guiding the search, another metrics, which prefers positive qualifications of positive examples, can be applied. This may lead to better results since performers' profiles are typically defined by positive rather than negative properties. Furthermore, another way of dealing with dependent attributes may also be considered, e.g., by incorporating a reasoning-component. In addition, alternative learning methods are interesting subjects of study. In particular, inductive logic programming [21] and mining association rules [13] seem to be interesting in this context. Alternative learning techniques will be an important subject for future research.

Besides possible enhancements of the mining procedure itself, many other interesting questions concerning *staff assignment mining* have arised, e.g., dealing with dependent staff assignment rules (i.e., the performer working on activity d should be always the same as the one who worked on a preceding activity c), and combining different mining perspectives (e.g., for credit amounts greater than

50000 € other agents are needed). Furthermore, for comprehensively supporting the process engineer, validation features for staff assignment rules also seem very useful in a respective tool.

We are currently working on an implementation of our approach as a plug-in for the ProM framework[2] [8]. ProM is a *process mining* tool where particular approaches can be incorporated as plug-ins. For further information on ProM, please refer to [8].

Though tests on real data sets are still required, we believe that the first steps are taken to mine the relations between the process and organisational structures.

References

1. zur Mühlen, M.: Organizational Management in Workflow Applications. In: Information Technology and Management, 5(3-4) (2004), 271–291
2. v.d. Aalst, W., Weijters, A.: Process Mining: A Research Agenda. In: Computers in Industry, 53(3) (2004), 231–244
3. v. d. Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., Weijters, A.: Workflow mining: A survey of issues and approaches. In: Data and Knowledge Engineering, 47(2) (2003), 237–267
4. v. d. Aalst, W., Song, M.: Mining Social Networks: Uncovering Interaction Patterns in Business Processes. In: BPM'04, Potsdam. Lecture Notes in Computer Science, Vol. 3080 (2004), 244–260
5. v. d. Aalst, W., Song, M.: Discovering Social Networks from Event Logs. BETA Working Paper Series, WP 116, Eindhoven University of Technology, The Netherlands (2004)
6. v. d. Aalst, W., Weijters, A., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. In: IEEE Transactions on Knowledge and Data Engineering, Vol. 16(9) (2004), 1128–1142
7. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.: Genetic Process Mining. In: 26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005). Lecture Notes in Computer Science, Vol. 3536. Springer Verlag (2005), 48–69
8. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds): 26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005). Lecture Notes in Computer Science, Vol. 3536 (2005), 444–454
9. Breslow, L., Aha, D.: Simplifying decision trees: a survey. In: Knowledge Engineering Review, 12(1) (1997), 1–40
10. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Mining Expressive Process Models by Clustering Workflow Traces. In: PAKDD, (2004), 52–62
11. Cook, J., Wolf, L.: Discovering Models of Software Processes from Event-Based Data. In: ACM Transactions on Software Engineering and Methodology, 7(3) (1998), 215-249

[2] ProM can be obtained from the web: www.processmining.org

12. Golani, M., Pinter, S.: Generating a Process Model from a Process Audit Log. In: Business Process Management. Lecture Notes in Computer Science, Vol. 2678 (2003), 136–151
13. Hand, D., Mannila, H., Smyth, P.: Priciples of Data Mining. MIT Press (2001)
14. Herbst, J.: A Machine Learning Approach to Workflow Management. In: ECML, (2000), 183–194
15. Mitchell, T.: Machine Learning. McGraw-Hill (1997)
16. Quinlan, J. R.: C4.5: Programs for Machine Learning Morgan Kaufmann Publishers, Inc. (1993)
17. Quinlan, J. R.: Learning Decision Tree Classifiers. In: ACM Computing Surveys, 28(1) (1996), 71–72
18. Reichert, M.; Wiedemuth–Catrinescu, U.; Rinderle, S.: Evolution of Access Control in Information Systems. In: Proc. Conf. EGP'04, Klagenfurt (2004), 100–114 (in German)
19. Rinderle, S.; Reichert, M.: On the Controlled Evolution of Access Rules in Information Systems. In: Proc. 13th Int'l Conf. on Cooperative Information Systems (CoopIS'05), Agia Napa, Cyprus (2005) (to appear)
20. Schimm, G.: Mining Exact Models of Concurrent Workflows. In: Process/Workflow Mining, Computers in Industry, 53 (3), Elsevier (2004), 265–281
21. Lavrac, N., Dzeroski, S.: Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York 53 (1994)
22. Weber, B., Reichert, M., Wild, W., Rinderle, S.: Balancing Flexibility and Security in Adaptive Process Management Systems. In: Proc. 13th Int'l Conf. on Cooperative Information Systems (CoopIS'05), Agia Napa, Cyprus (2005) (to appear)