Mining CPN Models

Discovering Process Models with Data from Event Logs

A. Rozinat, R.S. Mans, and W.M.P. van der Aalst

Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{a.rozinat,r.s.mans,w.m.p.v.d.aalst}@tm.tue.nl

Abstract. Process-aware information systems typically log events (e.g., in transaction logs or audit trails) related to the actual execution of business processes. Proper analysis of these execution logs can yield important knowledge that can help organizations to improve the quality of their services. Starting from a process model, which can be discovered by conventional process mining algorithms, we analyze how data attributes influence the choices made in the process based on past process executions using decision mining, or decision point analysis. In this paper we describe how the resulting model (including the discovered data dependencies) can be represented as a *Colored Petri Net* (CPN), and how further perspectives, such as the performance and organizational perspective, can be incorporated. We also present a *CPN Tools Export* implemented within the ProM framework. Using this plug-in simulation models in ProM obtained via a combination of various process mining techniques can be exported to CPN Tools. We believe that the combination of automatic discovery of process models using ProM and the simulation capabilities of CPN Tools offers an innovative way to improve business processes. The initially discovered process model describes reality better than most hand-crafted simulation models. Moreover, the simulation models are constructed in such a way that it is easy to explore various redesigns.

1 Introduction

Process mining techniques have proven to be a valuable tool in order to gain insight into how business processes are handled within organizations. Taking a set of real process executions (the so-called "event logs") as the starting point, these techniques can be used for *process discovery* and *conformance checking*. Process discovery [4, 6] can be used to automatically construct a process model reflecting the behavior that has been observed and recorded in the event log. Conformance checking [1, 10] can be used to compare the recorded behavior with some already existing process model to detect possible deviations. Both may serve as *input* for designing and improving business processes, e.g., conformance checking can be used to find problems in existing processes, and process discovery can be used as a starting point for process analysis and system configuration. While there are several process mining algorithms that deal with the control flow perspective of a business process [4] *less attention has been paid to how the value of a data attribute may affect the routing of a case.*

Most information systems (cf. WFM, ERP, CRM, SCM, and B2B systems) provide some kind of *event log* (also referred to as transaction log or audit trail) [4] where an event refers to a case (i.e., process instance) and an activity, and, in most systems, also a timestamp, a performer, and some additional data. Nevertheless, many process mining techniques only make use of the first two attributes in order to construct a process model which reflects the causal relations that have been observed among the activities. In this paper we start from a discovered process model (i.e., a model discovered by conventional process mining algorithms), and we try to enhance the model by integrating patterns that can be observed from data modifications, i.e., a *decision point analysis* [**?**] will be carried out in order to find out which properties of a case might lead to taking certain paths in the process. Colored Petri Nets (CPNs) [**?**] are then a suitable representation for the enhanced model because of their expressiveness and the good tool support provided through CPN Tools [**?**] (which, for example, has strong simulation capabilities). Furthermore, the hierarchy concept allows for the composition of CPN model in a modular way. The time concept and the availability of many probability distributions in CPN Tools allow for the modeling of performance aspects. Moreover, by introducing resource tokens also organizational and work distribution aspects can be modeled.

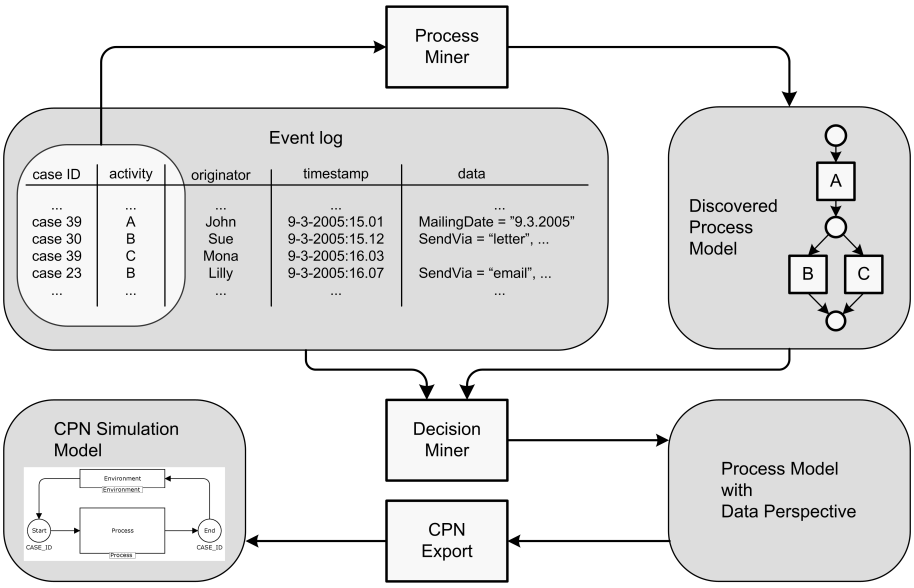


Fig. 1. The approach pursued in this paper

Figure 1 illustrates the overall approach. First of all, some process mining algorithm is used to discover a process model in terms of a Petri net (e.g., the α-algorithm [6]). Note that conventional process miners (e.g., based on the α-algorithm) only use the first two columns depicted in Figure 1. However, the event log may contain also

information about the people executing activities (cf. originator column), the timing of these activities (cf. timestamp column), and the data involved (cf. data column). In the next step we make use of the additional information, the data column to be precise. The Decision Miner uses this information to discover rules for taking alternative paths based on values of the data attributes present in the process. Finally, the process model including the data perspective is exported as a CPN simulation model. The CPN simulation model may be extended with additional information about time and resources. This information may be manually included or is extracted from the log based on the originator column and timestamp column.

To directly support the generation of a CPN simulation model for business processes we have implemented a *CPN Tools Export* plug-in in the context of the ProM framework[1], which offers a wide range of tools related to process mining and process analysis.

The paper is organized as follows. First, Section 2 introduces a simple example process that is used throughout the paper. Then, the decision mining approach is explained briefly in Section 3. Subsequently, we describe how a business process (including multiple perspectives) can be represented as a CPN in Section 4. Section 5 presents the CPN Tools export plug-in of the ProM framework. Finally, related work is discussed in Section 6, and the paper concludes by pointing out future research directions.

2 Running Example

As pointed out in Figure 1 the first step in the decision mining process is to obtain a process model without data through some classical *Process Miner*, e.g., a Petri net discovered using the α-algorithm. Figure 2(a) shows an event log in a schematic way, i.e., as a set of event traces. Note that this information can be extracted from the first two columns of the event log shown in Figure 1. Based on this information the α-algorithm automatically constructs the process model shown in Figure 2(b).

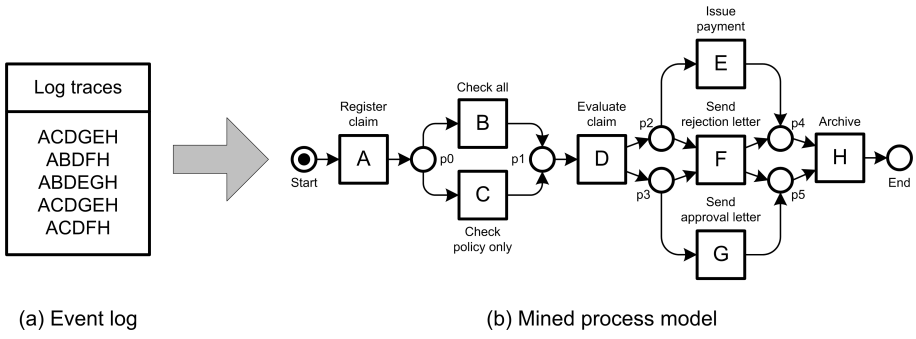


Fig. 2. Process mining phase

[1] Both documentation and software (including the source code) can be downloaded from *www.processmining.org*.

The example process used throughout the paper sketches the processing of a liability claim within an insurance company: first, some data related to the claim is registered (A), and then either a full check or a policy-only check is performed (B or C). Afterwards, the claim will be evaluated (D), and then it is either rejected (F) or approved (E and G). Finally, the case is archived and closed (H).

Now we have discovered the control flow perspective of the process. But the process execution log contains much more valuable information. In order to generate a simulation model that reflects as close as possible the process that has been observed, case data attributes, timestamps, and originator information can be analyzed to reveal characteristics related to the *data*, *performance*, and *organizational* perspectives. Figure 3 depicts a screenshot of the event log in MXML[2] format, and in the following we will have a closer look at it considering these perspectives.

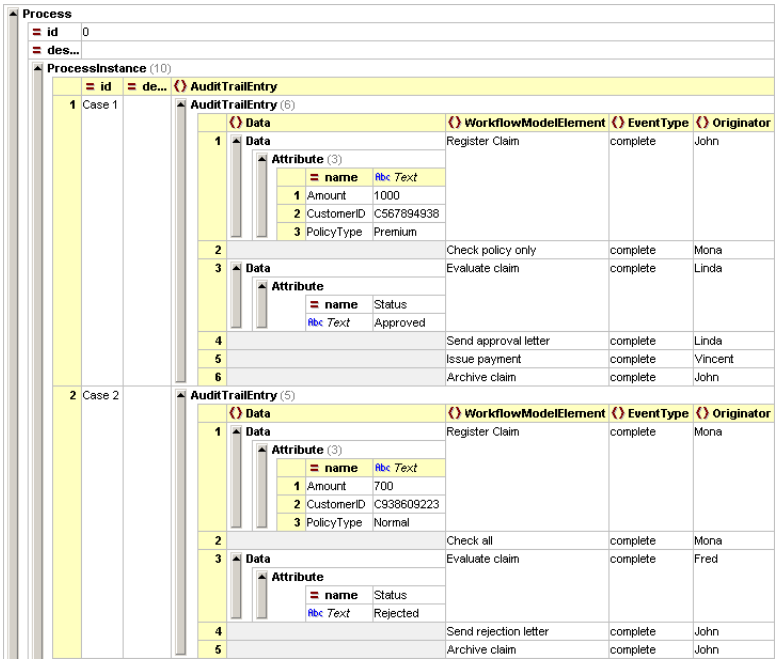


Fig. 3. Fragment of the example log in MXML format viewed using XML Spy

(a) *Data perspective*. Here a data item within an audit trail entry is interpreted as a case attribute that has been created, or modified. In Figure 3 one can observe that only activities *Register claim* and *Evaluate claim* have data items associated. During the execution of activity *Register claim* information about the amount of money

[2] Both the corresponding schema definition and the ProMimport framework, which converts logs from existing (commercial) process-aware information systems to the XML format used by ProM, can be downloaded from *www.processmining.org*.

involved (*Amount*), the corresponding customer (*CustomerID*), and the type of policy (*PolicyType*) are provided, while after handling the activity *Evaluate claim* the outcome of the evaluation is recorded (*Status*). Semantically, *Amount* is a numerical attribute, *CustomerID* is an attribute which is unique for each customer, and both *PolicyType* and *Status* are enumeration types (being either "Normal" or "Premium", or either "Approved" or "Rejected", respectively).

(b) *Performance perspective*. In the example, for simplicity, activities are considered as being atomic and carry no time information. However, information systems dealing with processes typically log events on a more fine-grained level, e.g., they record *schedule*, *start*, and *complete* events (including timestamps) for each activity that has been first activated and then executed. Thus, time information can be used to infer, e.g., activity durations, or the arrival rate of newly started cases. Furthermore, the frequency of alternative paths represents quantitative information that is implicitly contained in the event log. For example, the event log shown in Figure 3 contains 10 process instances, of which 7 executed activity *Check policy only* and only 3 performed the full check procedure *Check all*.

(c) *Organizational perspective*. In Figure 3 one can observe that each event carries information about the resource that executed the activity. In the insurance handling example process 7 different persons have worked together: Howard, Fred, Mona, Vincent, Robert, Linda, and John.

As illustrated in Figure 1, the discovered process model and the detailed log are the starting point for the *Decision Miner*, which analyzes the data perspective of the process in order to discover data dependencies that influence the routing of a case. The idea of decision mining is briefly explained in the next section (see [**?**] for further details). The Decision Miner constructs a simulation model incorporating the data perspective and passes this on to the *CPN Export* plug-in. However, in addition to the control-flow and data perspective the simulation model may also contain information about resources, probabilities, and time (i.e., the performance and organizational perspectives).[3] The representation of all these perspectives in terms of a CPN model and the configuration possibilities of the CPN Tools Export in ProM are described in Section 4 and Section 5.

3 Decision Mining

In order to analyze the choices in a business process we first need to identify those parts of the model where the process splits into alternative branches, also called *decision points*. Based on data attributes associated to the cases in the event log we subsequently want to find rules for following one route or the other.

In terms of a Petri net, a decision point corresponds to a place with multiple outgoing arcs. Since a token can only be consumed by one of the transitions connected to

[3] These perspectives can be added by hand or through additional process mining techniques. We are currently working on integrating the information from various plug-ins, focusing on integrating the performance and organizational perspectives with the information from the Decision Miner.

these arcs, alternative paths may be taken during the execution of a process instance. The process model in Figure 2(b) exhibits three such decision points: *p0* (if there is a token, either B or C can be performed), *p2* (if there is a token, either E or F can be executed) and *p3* (if there is a token, either F or G may be carried out). In order to analyze the choices that were made in past process executions, we need to find out which alternative branch was taken by a certain process instance. Therefore, the set of possible decisions must be described with respect to the event log. Starting from the identification of a choice construct in the process model a decision can be detected if the execution of an activity in the respective alternative branch of the model has been observed, which requires a mapping from that activity to its "occurrence footprint" in the event log. So, if a process instance contains the given "footprint", this means that there was a decision for the associated alternative path in the process. For simplicity we examine the occurrence of the *first* activity per alternative branch in order to classify the possible decisions. However, in order to make decision mining operational for real-life business processes several challenges posed by, for example, *invisible activities*, *duplicate activities*, and *loops* need to be met. We refer the interested reader to our technical report [**?**], where these issues are addressed in detail.

After identifying a decision point in a business process and classifying the decisions of the process instances in the log, the next step is to determine whether this decision might be influenced by case data, i.e., whether cases with certain properties typically follow a specific route. The idea is to convert every decision point into a *classification problem* [**?**,**?**,**?**], where the *classes* are the different decisions that can be made. As training examples we use the process instances in the log (for which it is already known which alternative path they followed with respect to the decision point). The attributes to be analyzed are the case attributes contained in the log, and we assume that all attributes that have been written *before* the choice construct under consideration are relevant for the routing of a case at that point[4]. In order to solve such a classification problem, various algorithms are available [**?**,**?**]. We decided to use an algorithm based on decision trees (i.e., the C4.5 algorithm [**?**]). Decision trees are a popular tool for inductive inference and the corresponding algorithms have been extended in various ways to improve practical applicability. For example, they are able to deal with continuous-valued attributes, missing attribute values, and they include effective methods to avoid *over-fitting* the data (i.e., that the tree is too much tailored towards the particular training examples).

Using decision point analysis we can extract knowledge about decision rules as shown in Figure 4. Each of the three discovered decision points corresponds to one of the choices in the running example. With respect to decision point *p0* the extensive check (activity B) is only performed if the *Amount* is greater than 500 and the *PolicyType* is "Normal", whereas a simpler coverage check (activity C) is sufficient if the *Amount* is smaller than or equal to 500, or the *PolicyType* is "Premium" (which may be due to certain guarantees given by "Premium" member corporations). The two choices at decision point *p2* and *p3* are both guided by the *Status* attribute, which is the outcome of the evaluation activity (activity D).

[4] We also allow the user to set other scoping rules, e.g., only the data set in a directly preceding activity, or all case data including the data that is set later.

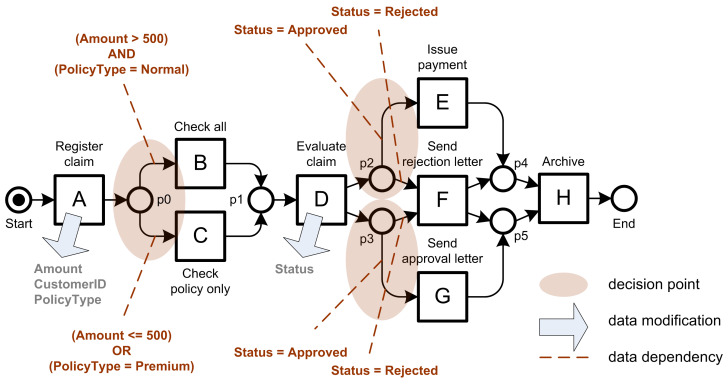


Fig. 4. Enhanced process model

Now that we have discovered a model of the control-flow and data perspective of the example process, we describe how this information (and information about the performance and organizational perspective) can be represented in a CPN model (Section 4), and show how such a CPN simulation model can be generated in ProM (Section 5).

4 CPN Simulation Models

Since we want to make use of the simulation facilities of CPN Tools, we provide the actual process model together with a simulation environment. The top-level page in the hierarchical CPN model is shown in Figure 5. For each process model this page will look identical; the environment generates cases and puts them into the *Start* place, and removes those that have finally reached the *End* place. We assume that the discovered process represented by the sub-page *Process* is sound, i.e., any case that enters the sub-page via place *Start* leaves the sub-page via place *End*.

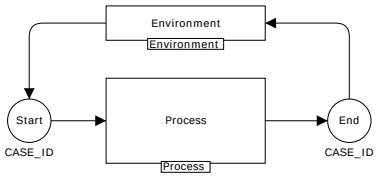


Fig. 5. Overview page

Figure 6 depicts the simulation environment in more detail. One can observe that the CASE_ID is an integer which is simply incremented for each generated process instance. For the data perspective, a separate token containing the case ID and a record of case attributes (defined via the DATA color set) is created and initialized.

The place *Case data* is modeled as a fusion place as activities may need to inspect or modify attribute values on a different page in the hierarchical model. Furthermore, the *Resources* fusion place contains the available resources for the process, and therefore determines the environment from an organizational perspective. Finally, each time a token is put back in the *next case ID* place a time delay[5] is added to it, which is used to steer the generation of new cases. In Figure 6 a constant time delay of 3 implements that every 3 time units a new case arrives. Note that the inter-arrival times may also be sampled from some probability distribution discovered by ProM.

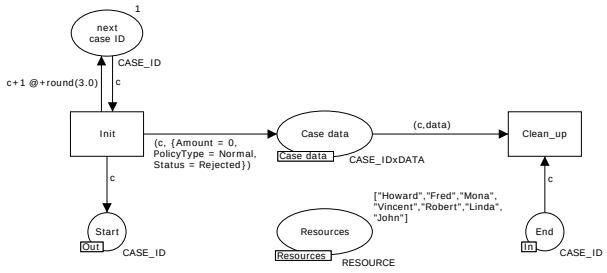


Fig. 6. Environment page

Figure 7 shows the sub-page containing the actual process model, which looks exactly like the original, low-level Petri net. Note that the tokens routed from the *Start* to the *End* place are of type CASE_ID, so that tokens belonging to different instances are not mixed up.

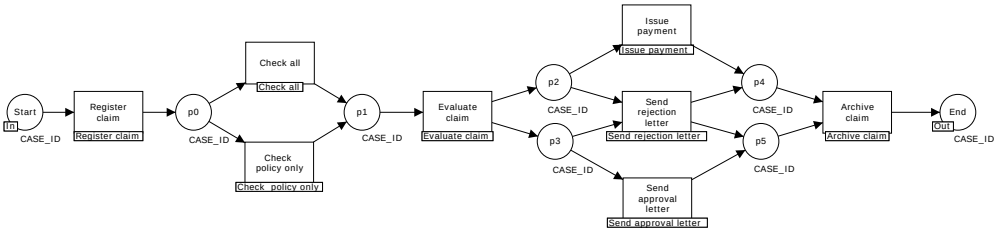


Fig. 7. Process page

Every activity on the process page has its own sub-page containing the actual simulation information. Depending on the simulation configuration these activity sub-pages

[5] Note that in our simulation model the time delay is always attached to an arc (depending on the token that should be delayed) rather than using the time delay of a transition in order to reduce side effects on other tokens that should actually not be delayed (such as the *Case data* token).

may look very different. In the following sub sections we will present how simulation information from several dimensions can be represented in terms of a CPN sub-page.

4.1 Data

Taking the enhanced model from Figure 4 as the starting point, we now want to incorporate the discovered data dependencies in the simulation model. The discovered decision rules are based on attributes provided by activity *Register claim* and *Evaluate claim* respectively. Since the attribute *CustomerID* is not involved in the discovered rules, we discard it from the process model and define process-specific data types for each of the remaining attributes (i.e., AMOUNT, POLICYTYPE, and STATUS).

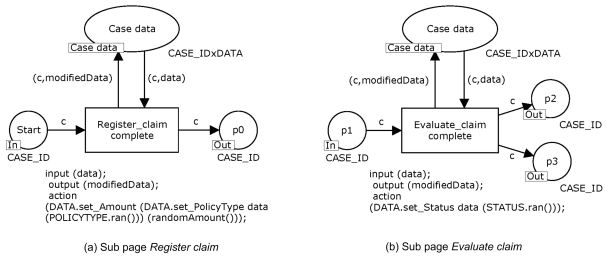


Fig. 8. Writing data items using random values

Figure 8 shows how the provision of case data can be modeled using random values. While a random value for a nominal attribute can be generated by applying the *ran()* function directly to the color set[6] a dedicated random function is needed for numeric attributes. In the action part of transition *Register claim* function *POLICYTYPE.ran()* is used to select the policy type ("Normal" or "Premium") and a dedicated function *randomAmount()* is used to set the amount. In this case, the amount is sampled from a uniform distribution generating a value between the lowest and the highest attribute value observed in the event log. However, many other settings are possible.

Figure 9 shows how the discovered data dependencies can then be modeled with the help of transition guards. If the transition is enabled from a control-flow perspective, it additionally needs to satisfy the given guard condition in order to be fired.

4.2 Time

Although there is no time information in the example event log, we want to include the time dimension in our simulation model. We can, for example, say that—in contrast to the policy-only check, which takes between 3 and 8 time units—the full check procedure needs between 9 and 17 time units to complete. Furthermore, the time between the

[6] Note that the *ran()* function can only be used for enumerated color sets with less than 100 elements.

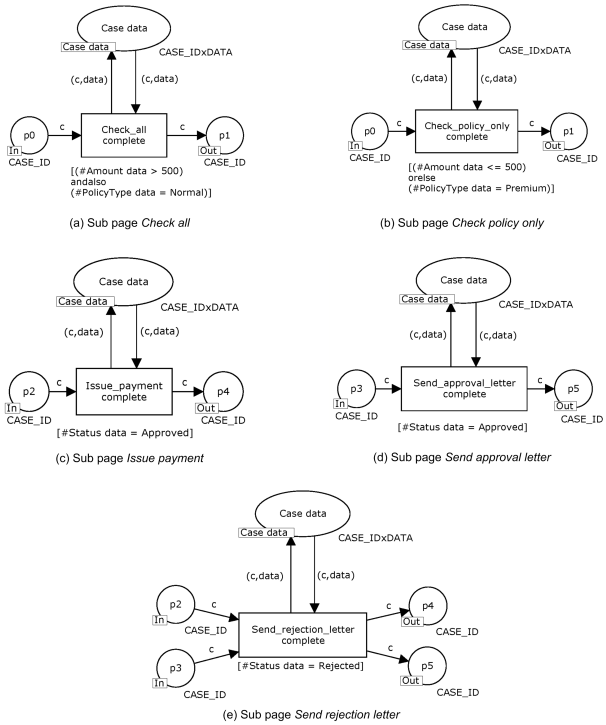


Fig. 9. Modeling data dependencies using transition guards

point where the activity *could have been started* (i.e., all required previous activities were completed) and the point where someone *actually starts* working on it may vary from 3 to 5 time units. Whereas the sub-page shown in Figure 9(a) models the activity *Check all* in an atomic way, one can distinguish between *schedule*, *start*, and *complete* transitions in order to incorporate the *waiting time* and *execution time* of this activity. Figure 10 shows three ways to model this for activity *Check all*.

In Figure 10(a) only the execution time of the activity is modeled. When transition *Check_all start* is fired, a token is produced with the indicated time delay. Similar to the case generation scheme in Figure 6, the token will remain between 9 and 17 time units in the place E (i.e., the activity is in the state *Executing*) before transition *Check_all complete* may fire.

In Figure 10(b) both the execution time and the waiting time are explicitly modeled. Analogously to the execution time, the waiting time is realized by a time delay that forces the token to reside in place W (i.e., the activity is in the state *Waiting*) between 3 and 5 time units before transition *Check_all start* may fire.

In Figure 10(c) the sum of the waiting time and the execution time is modeled. This may be useful if no information is available about the actual start of an activity (i.e., only the time when it becomes enabled and when it is finished is known).

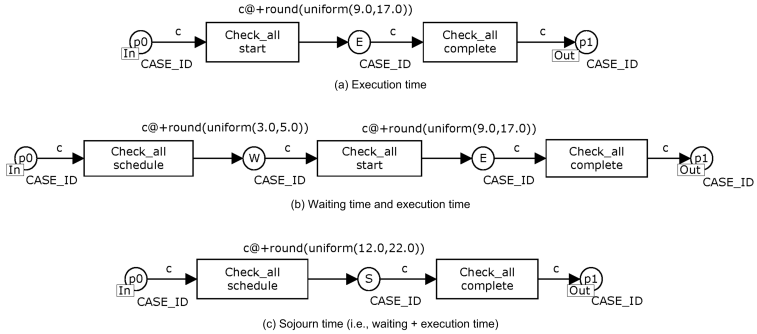


Fig. 10. Different variants of modeling time on sub-page *Check all* depending on the event types (i.e., schedule, start, and complete) present in the log

4.3 Resources

In order to gain insight into the organizational perspective we analyze the event log shown in Figure 3 with the *Social Network miner* of ProM. Figure 11 shows a social network generated based on the metric *similar work* [3], i.e., two people are linked in the social network if they execute similar activities. The more similar their execution profiles are the stronger their relationship. In the social network depicted one can observe that Vincent and Howard execute a set of activities which is disjoint from those executed by all other employees. More precisely, they only execute the activity *Issue payment* and, therefore, might work, e.g., in the *Finance* department of the insurance company. Furthermore, the work of Fred and Linda seems to be more similar to each other than to the work of the other three people. Having a closer look at the event log again reveals that they are the only people performing the *Evaluate claim* activity, although they also execute other activities, (such as *Send rejection letter* and *Archive claim*). This may be due to the fact that the activity *Evaluate claim* requires some *Manager* role, whereas all the remaining activities can be performed by people having a *Clerk* role.

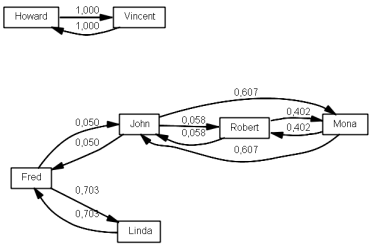


Fig. 11. Social network based on the metric "similar work" [3]

A simple way to incorporate this knowledge in our simulation model is to create three groups of resources (Finance = {Howard, Vincent}, Manager = {Fred, Linda}, Clerk = {Fred, Linda, John, Robert, Mona}) and to specify for each activity which kind of resource is required (if no particular group has been specified for an activity, it can be performed by any resource). Figure 12 depicts how the fact that activity *Evaluate claim* requires the role *Manger* is modeled in the corresponding CPN model. The role is modeled as a separate color set MANAGER, which contains only "Lisa" and "Fred". Because the variable *g1* is of type MANAGER, only the resource token "Lisa" or "Fred" can be consumed by transition *Evaluate_claim start*. As soon as transition *Evaluate_claim start* is fired, the corresponding resource token resides in the place *E*, i.e., it is not available for concurrent executions of further activities, until transition *Evaluate_claim complete* fires and puts the token back.

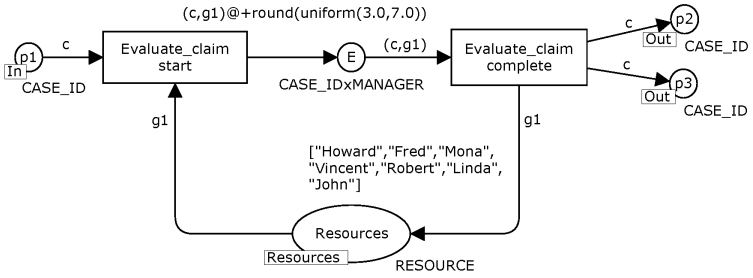


Fig. 12. Sub-page *Evaluate claim* including resource modeling

4.4 Probabilities and Frequencies

In addition to the modeling of data, time and resource information, one may also be interested in stochastic aspects. Hence, these aspects also need to be incorporated in the CPN model. Figure 13 shows how often each arc in the model has been used, determined through the log replay analysis carried out by the *Conformance Checker* in ProM[7]. Looking at the first choice it becomes clear that activity *Check policy only* has been executed 7 (out of 10) times and activity *Check all* was performed only 3 times. Similarly, activity *Send rejection letter* happened for 4 (out of 10) cases, while in 6 cases both activity *Send approval letter* and activity *Issue payment* were executed.

In order to incorporate frequencies of alternative paths in the simulation model we use two different approaches, depending on the nature of the choice.

Simple choice The first choice construct in the example model is considered to be a so-called *simple choice* as it is only represented by one single place. We can model

[7] Note that the place names and the markup of the choices have been added to the diagnostic picture obtained from ProM for explanation purposes.

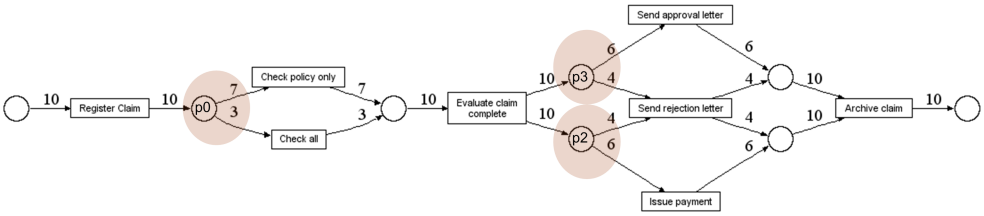


Fig. 13. Frequencies of alternative paths in the example model

such a simple choice using a probability token that is shared among all the activities involved in this choice via a fusion place.
Figure 14 shows this solution for the choice at place *p0*. Both sub-pages *Check all* and *Check policy only* contain a fusion place *p0_Probability* that initially contains a token with the value 0, but after each firing of either transition *Check_all start* or transition *Check_policy_only start* a random value between 0 and 100 is generated. Because of the guard condition, the decision at the place *p0* is then determined for each case according to the current value of the token in place *p0_Probability*. For example, the transition *Check_all start* needs to bind the variable *prob* to a value greater than or equal to 70 in order to be enabled, which will only happen in 30% of the cases.

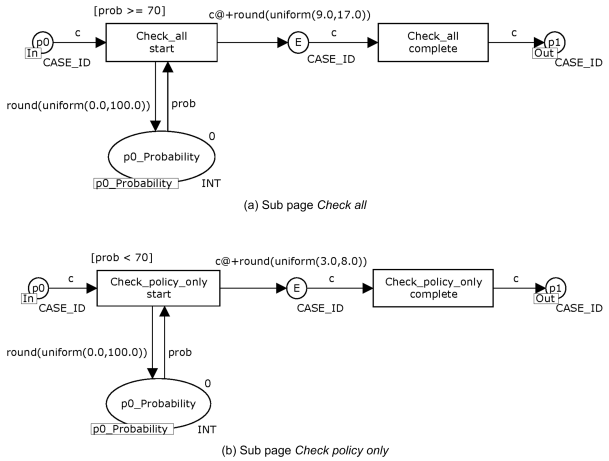


Fig. 14. Using a probability token for simple choices

Dependent choices The second choice construct in the example model actually consists of two *dependent choices*[8] (i.e., the choices represented by places *p2* and *p3*)

[8] Similar to the Decision Miner we consider each place as a choice (or decision point) if it contains more than one outgoing arc (cf. Figure 4).

that need to be coordinated in order to either approve or reject a claim. It is clear that two dependent choices cannot be steered properly by two independently generated probability tokens, because the process model will deadlock as soon as the values of the probability tokens indicate contrasting decisions (e.g., the probability token in *p2* indicates a reject while the other probability token in *p3* suggests to approve the claim).
Figure 15 shows a solution for modeling the dependent choices at place *p2* and *p3*. The idea is to increase the likelihood of choosing a certain activity through activity duplication (because during simulation any enabled transition will be fired with an equal probability). This way, the observed relative frequency[9] of the transitions involved in the dependent choices can be incorporated in the simulation model. Figure 15(a) shows an intermediate sub-page for activity *Issue payment*, where three substitution transitions *Issue payment* point to different instances of the same sub-page *Issue payment* (i.e., the actual sub-page is only modeled once). Figures 15(b) and (c) show similar intermediate sub-pages for the activities *Send approval letter* (also duplicated three times) and *Send rejection letter* (duplicated twice).

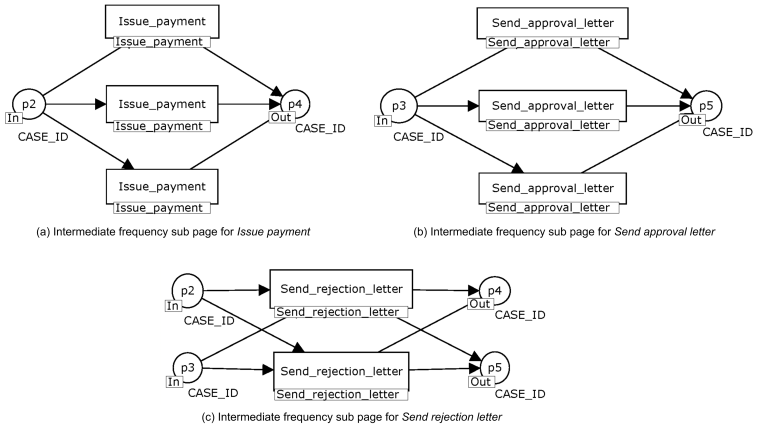


Fig. 15. Modeling dependent choices via activity duplication

4.5 Logging and Monitoring Simulation Runs

The CPN models described in this section deal with time, resources, and data. When running a simulation in CPN Tools we are interested in statistics (e.g. average, variance, minimum, and maximum) related to the utilization of resources and the throughput times of cases during the simulation run.

[9] In order to obtain the relative frequency, the absolute frequency is divided by the greatest common divisor (i.e., $6/2 = 3$ and $4/2 = 2$).

In the case that resources have been defined for the process all the available resources are contained in a *Resources* fusion place, which is located on the *Environment* page and on every activity sub-page. For obtaining statistics about the resource utilization during the simulation we can simply define a *marking size monitor* for this Resources fusion place, which records the number of available resources plus the current time (and step) as soon as a resource becomes (un-)available.

Furthermore, when the timing perspective is enabled, tokens are created with a timestamp. In order to calculate the throughput time for each case, we record the timestamp of its creation together with the case ID token that is routed through the process. This way, the current run time of a case can be easily determined at each stage in the process via some custom monitor. For the throughput time monitor, a *data collector monitor* has been defined for the *Clean up* transition on the *Environment* page (cf. Figure 6), which simply calculates the difference between the current model time and the start time of a case[10], and records the throughput time, the end time and end step for each case.

Moreover, we want to generate process execution logs for the business process in the CPN simulation model. This can be very useful for, e.g., the creation of artificial logs that are needed to evaluate the performance of process mining algorithms.

For each firing of a transition on an activity sub-page an event is logged, which includes case ID, the type of transition (i.e., *schedule*, *start*, or *complete*), current timestamp, originator, and additional data (each if available). For generating these process execution logs we use the logging functions that have been described in [**?**]. However, in contrast to [**?**]—where the *input/output/action* inscriptions of transitions have to be modified to invoke these logging functions—we decided to use *user defined monitors* in order to clearly separate the logging from the rest of the simulation model.

Figure 16 shows the monitor that has been defined for the *complete* transition of *Evaluate claim*. While firing this transition a random value is generated for the *Status* attribute and it may only be performed by people having the *Manager* role (cf. figures 8(b) and 12). Each time *Evaluate_claim complete* fires, the monitor does the following. The *Predicate* function checks whether all variables that belong to the transition are bounded. Then the *Observer* function is executed. This function extracts information from the net that can be passed to the *Action* function. Here, we extract the case ID (variable *c*), the originator (variable *g1*) and the data that has been modified (variable *modifiedData*), which will be passed to the *Action* function (the original data (variable *data*) is also extracted but does not need to be passed to the *Action* function). In the *Action* function the passed values are used to create the corresponding Audit Trail Entry that needs to be written to the log file of the case. For this we use the function *addATE* that has been described in [**?**]. In Figure 16 the *addATE* function writes an Audit Trail Entry to the log file of the case ID that is represented by variable *f*. For the Audit Trail Entry itself, the *name* of the transition is recorded,

[10] Because the type of the current model time is infinite integer and in order to not lose precision when calculating the difference between the current model time and the start time of a case, the model time is mapped onto a STRING value, i.e., color set START_TIME is of type STRING and is used to encode infinite integers.

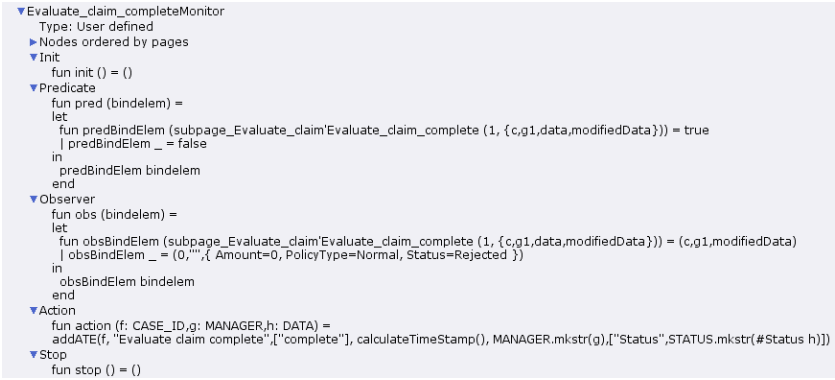


Fig. 16. User defined monitor for *Evaluate_claim complete*

the corresponding *event type*, the *timestamp* at which the transition has been executed, which *manager* completed the activity, and finally the *case attribute* that has been changed.

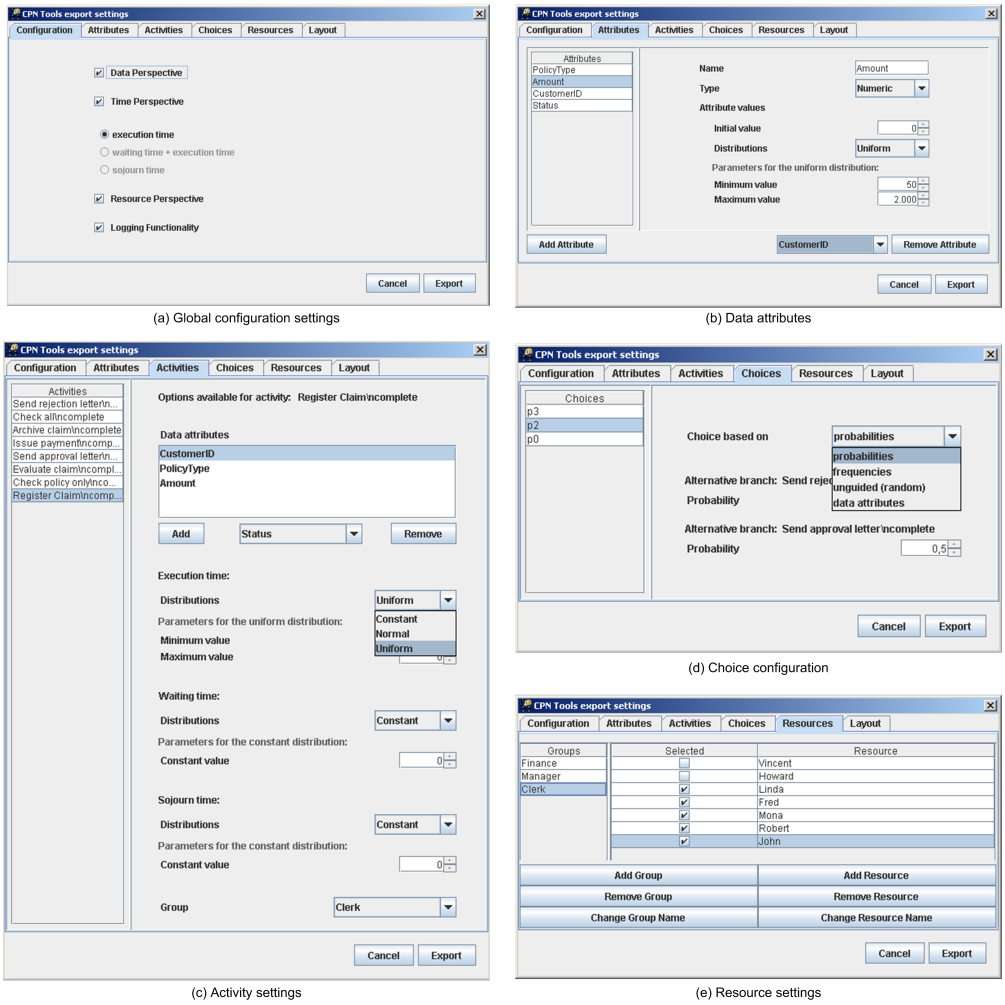
Note that the *Init* and the *Stop* functions are not used because it is not necessary to initialize or terminate the monitor. However, before it is even possible to write to a log file, we first need to open a log file for each case. This is ensured by defining a monitor for the *Init* transition on the *Environment* page that initializes every case. In the *Action* function of that monitor the function *createCaseFile* is called, which opens a log file for every case.

5 Exporting CPN Models from ProM

We are able to generate CPN simulation models as presented in the previous section using the *CPN Tools 2.0 Export* plug-in in the ProM framework[11]. It either accepts a simulation model that has been provided by another plug-in in the framework, or a simple, low-level Petri net (in which case an empty simulation model is created and filled with default values). Before the actual export takes place, it allows for the manipulation of the simulation information in the model. As illustrated in Figure 1, we have discovered a process model including the data perspective (provided by the *Decision Miner*), and we can now manually integrate information about further dimensions, such as the performance and organizational dimension.

Figure 17(a) shows the global Configuration settings, where the user can choose which dimensions should be included in the generated simulation model. In fact, although the relevant simulation information may be provided, it will be ignored if the corresponding configuration option was not chosen. Note that, since the waiting time of an activity typically results from the unavailability of resources, explicit modeling of a resource scheme such as in Figure 12 is in conflict with modeling the time dimension

[11] Note that the layout of the generated models was slightly adjusted in order to improve the readability.



(a) Global configuration settings

(b) Data attributes

(c) Activity settings

(d) Choice configuration

(e) Resource settings

Fig. 17. CPN Tools export settings

including waiting time (cf. figures 10(b) and (c)). Therefore, only the execution time option is available if the Resource dimension is selected.

Figure 17(b) depicts the Attributes settings of the process. New data attributes can be provided by specifying their name, type (nominal or numeric), possible values (a list of String values for a nominal attribute, and some distribution for a numeric one), and initial value. However, the available data attributes for the example process were already discovered by the Decision Miner. We might only want to delete the *CustomerID* attribute, since it is not involved in any of the discovered data dependencies.

In Figure 17(c), a screenshot of the Activity settings for activity *Register claim* is displayed. In this view, the provided data attributes, the execution time, waiting time,

sojourn time, and the required resource group may be specified for each of the activities in the process. The data attributes were already provided by the Decision Miner, but we can assign some execution time to each activity (the probability distributions currently supported by the CPN export plug-in are the constant, uniform, and normal distribution), and choose the suitable group of resources from the list of groups available in the process.

Figure 17(d) shows the Choice configuration view, where the user can determine for each decision point in the process whether it should be based on either probabilities or frequencies (cf. Section 4.4), or on data attributes, or whether it should not be guided by the simulation model (one of the alternative paths is then randomly chosen by CPN Tools). In Figure 17(d) the probability settings are displayed. For every alternative branch a probability may be provided between 0.0 and 1.0. Before the actual export takes place, each value is normalized by the sum of all specified values, and if the values sum up to 0.0, default values (i.e., equal probabilities for each alternative branch) are used. As discussed in Section 4.4, for dependent choices one should specify a relative frequency value instead. Finally, we can provide a dependency value based on data attributes (in the case of our example process the discovered dependency has already been filled in by the Decision Miner). In the current version of the export plug-in this dependency value is simply a String containing the condition to be placed in the transition guard of the corresponding transition.

In Figure 17(e) the Resources settings are depicted. Here, one can add groups and resources, and assign resources to groups. This way the CPN Tools 2.0 Export plug-in also supports the export of information about resources. This information is then used to create the sub-pages shown earlier.

6 Related Work

The work reported in this paper is related to earlier work on process mining, i.e., discovering a process model based on some event log. The idea of applying process mining in the context of workflow management was first introduced in [7]. Cook and Wolf have investigated similar issues in the context of software engineering processes using different approaches [8]. Herbst and Karagiannis also address the issue of process mining in the context of workflow management using an inductive approach [9]. They use stochastic task graphs as an intermediate representation and generate a workflow model described in the ADONIS modeling language. Then there are several variants of the α algorithm [6, 11]. In [6] it is shown that this algorithm can be proven to be correct for a large class of processes. In [11] a heuristic approach using rather simple metrics is used to construct so-called "dependency/frequency tables" and "dependency/frequency graphs". This is used as input for the α algorithm. As a result it is possible to tackle the problem of noise. For more information on process mining we refer to a special issue of Computers in Industry on process mining [5] and a survey paper [4]. However, as far as we know, this is the first attempt to mine process models including other dimensions, such as data.

In [**?**] the authors present a translation of Protos simulation models to CPN Tools. In addition, three types of data collector monitors (measuring the total flow time per

case, the waiting time per task, and the resource availability/utilization per resource type), and configuration features enabling the dynamic elimination of unnecessary parts of the process model are generated. Besides the work in [**?**], we are not aware of further attempts to export business process models to CPN Tools. The work reported in this paper has a different starting point as it is not limited by the simulation information present in a Protos model, but aims at discovering the process characteristics to be simulated from the event logs of real process executions.

7 Future Work

Future work includes the refinement of the generated CPN models and the further exploitation of the features already present in CPN Tools. For example, a more realistic resource modeling scheme may allow for the specification of a working scheme per resource (e.g., whether the person works half-time or full-time) and include different allocation mechanisms, and we plan to support all random distributions that are available in CPN Tools in the near future. Moreover, the discovery of further perspectives of a business process will be integrated in the mined process models. Currently, we are able to discover data dependencies via the Decision Miner in ProM. But existing plug-ins in ProM will deliver also time-related characteristics of a process (such as the case arrival scheme, and execution and waiting times) and frequencies of alternative paths, or organizational characteristics (such as the roles of the employees involved in the process). All these different pieces of aggregate information (discovered from the event log) need then to be combined in one holistic simulation model, which may be exported to CPN Tools, or, e.g., translated to an executable YAWL model [2]. Note that a YAWL model can be used to enact a business process using the YAWL workflow engine. For enactment all perspectives play a role and need to be taken into account. Hence, successfully exporting to YAWL is another interesting test case for the mining of process models with data and resource information.

Acknowledgements

This research is supported by EIT and the IOP program of the Dutch Ministry of Economic Affairs. The authors would also like to thank Ton Weijters, Boudewijn van Dongen, Ana Karla Alves de Medeiros, Minseok Song, Laura Maruster, Christian Günther, Eric Verbeek, Monique Jansen-Vullers, Hajo Reijers, Michael Rosemann, Huub de Beer, Peter van den Brand, et al. for their on-going work on process mining techniques. We would also like to thank Lisa Wells and Kurt Jensen for their support in using CPN Tools.

References

1. W.M.P. van der Aalst. Business Alignment: Using Process Mining as a Tool for Delta Analysis. In J. Grundspenkis and M. Kirikova, editors, *Proceedings of the 5th Workshop on Business Process Modeling, Development and Support (BPMDS'04)*, volume 2 of *Caise'04 Workshops*, pages 138–145. Riga Technical University, Latvia, 2004.

2. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. A.K. Alves de Medeiros and C.W. Guenther. Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms. In Kurt Jensen, editor, *Proceedings of the Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 177–190, 2005.
10. F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, and H.M.W. Verbeek. Protos2CPN: Using Colored Petri Nets for Configuring and Testing Business Processes. In *Submitted to the Seventh Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, 2006.
11. 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.
12. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* Springer-Verlag, 1997.
13. T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
14. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
15. 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, *Business Process Management 2005 Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, Berlin, 2006.
16. A. Rozinat and W.M.P. van der Aalst. Decision Mining in Business Processes. BPM Center Report BPM-06-10, BPMcenter.org, 2006.
17. A. Vinter Ratzer, L. Wells, H. M. Lassen, M. Laursen, J. F. Qvortrup, M. S. Stissing, M. Westergaard, S. Christensen, and K. Jensen. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. In W.M.P. van der Aalst and E. Best, editors, *Applications and Theory of Petri Nets 2003: 24th International Conference, ICATPN 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 450–462. Springer Verlag, 2003.
18. 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.
19. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition.* Morgan Kaufmann, 2005.