Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Discovering Colored Petri Nets from Event Logs

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

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

Received: date / Revised version: date

Abstract. Process-aware information systems typically log events (e.g., in transaction logs or audit trails) related to the actual execution of business processes. Analysis of these execution logs may reveal 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, also referred to as 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` plug-in 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 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 has proven to be a valuable approach that provides new and objective insights into the way business processes are really 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 [5,7] can be used to automatically construct a process model reflecting the behavior that has been observed and recorded in the event log. Conformance checking [1,22] 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 [5] *less attention has been paid to how data attributes affect the routing of a case*. Classical process mining approaches consider all choices to be non-deterministic. The approach presented in this paper investigates how values of data attributes influence particular choices in the model.

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) [5] where an event refers to a case (i.e., process instance) and an activity, and, in most systems, also a time stamp, a performer, and some additional data. Nevertheless, many process mining techniques only make use of the first two attributes to construct a process model which reflects the causal relations that were 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* [24] will be carried out to find out which properties (i.e., valuations of data attributes) of a case might lead to taking

⋆ Supported by the IOP program of the Dutch Ministry of Economic Affairs.
⋆⋆ Supported by the Technology Foundation STW.

certain paths in the process. Colored Petri Nets (CPNs) [16,17] are used as a representation for the enhanced model because of their expressiveness and the good tool support provided through CPN Tools [27] (for example, we will show how CPN Tools enables simulation-based performance analysis of the discovered model). Furthermore, the hierarchy concept allows for the composition of a 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.

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 [7]). Note that conventional process mining techniques (e.g., based on the α-algorithm) only use the first two columns of the event log depicted in Figure 1. However, the event log may also contain information about the people performing activities (cf. originator column), the timing of these activities (cf. time stamp 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. The enhanced 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 time stamp column and originator column. Note that in this paper we focus on the CPN representation of the enhanced model rather than on the discovery techniques for the different perspectives. Therefore, decision mining is only one example of how a model can be enhanced by extracting additional information from the log. Many other techniques (e.g., role discovery [3], or the extraction of timing information as described in [15]) are possible. Finally, the process model including the data (and time and resource) perspective is exported as a CPN model.

We argue that *the discovered process model describes reality better than most hand-crafted simulation models* because it is based on objective information rather than on perceptions of people (i.e., there is no modeler bias). Of course, such a generated model does not make domain knowledge and modeling expertise obsolete. However, in this paper we propose a method that—in contrast to the manual creation of a model—*can be easily repeated in an iterative manner as soon as the process changes.* No modeling efforts are needed to generate an initial model, which can be further evaluated, and potentially modified.

To directly support the generation of a CPN model for business processes we have implemented a `CPN Tools 2.0 Export` plug-in (in the remainder of this paper referred to as `CPN Export` plug-in) in the context of the ProM framework[1]. The ProM framework offers a wide range of tools related to process mining and process analysis, and the `CPN Export` plug-in in ProM fully automatically generates the CPN models discovered using process mining. Note that we have applied our process mining techniques to many real-life logs from, e.g., hospitals, banks, municipalities etc. (see [4,25] for two examples). For structured processes this works well. However, for more chaotic processes it is difficult to produce models that are easy to interpret and analyze. For some of the case studies we constructed CPN models. However, a discussion of these case studies is outside the scope of this paper.

The paper is organized as follows. First, related work is discussed in Section 2. Then, Section 3 introduces a simple example process that is used throughout the paper. Afterwards, the decision mining approach is explained briefly in Section 4. In Section 5, we describe how a business process (including multiple perspectives) can be represented as a CPN. Then, Section 6 presents the `CPN Export` plug-in of the ProM framework, and Section 7 shows how the generated CPN models can be simulated and analyzed in CPN Tools. Finally, the paper concludes by discussing directions for future research.

2 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 [8]. Cook and Wolf have investigated similar issues in the context of software engineering processes using different approaches [9]. Herbst and Karagiannis also address the issue of process mining in the context of workflow management using an inductive approach [14]. They use stochastic task graphs as an intermediate representation and generate a workflow model described in the ADONIS modeling language. Alternatively, there are several variants of the α algorithm [7,28]. In [7] it is shown that this algorithm can be proven to be correct for a large class of processes. In [28] 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 [6] and a survey paper [5]. However, as far as we know, this is the first attempt to mine process models including other dimensions, such as data. (Note that [3] only considers the social network in isolation and does not use it to provide an integrated view.)

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

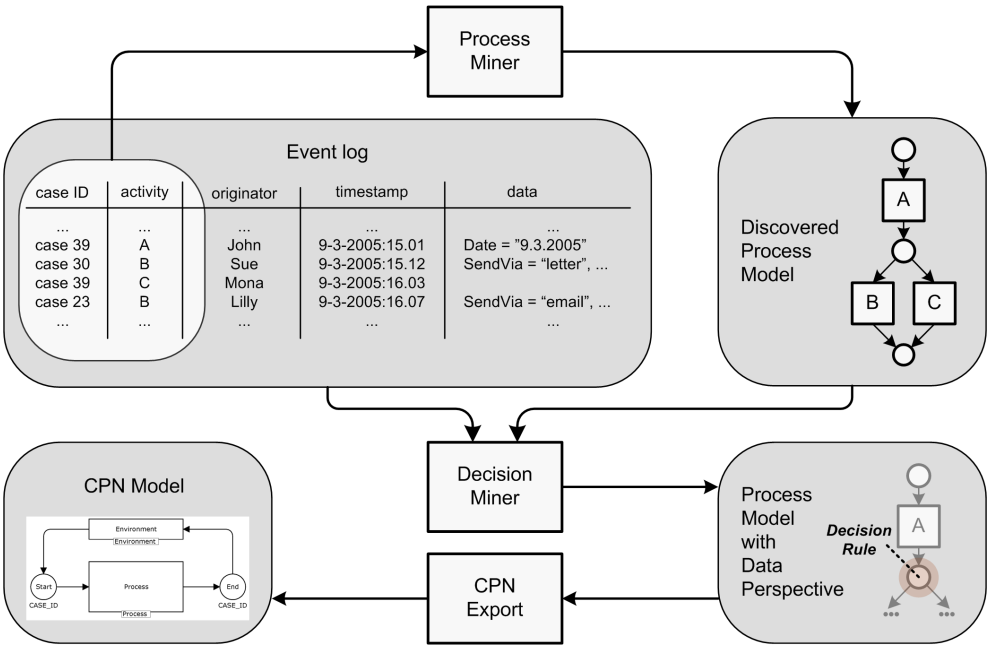


Fig. 1. The approach described in this paper

Our work on decision mining [24,23] (which we build upon in this paper) is closely related to [12], in which the authors describe the architecture of the *Business Process Intelligence* (BPI) tool suite situated on top of the *HP Process Manager* (HPPM). Whereas they outline the use of data mining techniques for process behavior analysis in a broader scope, we show how a decision point analysis can be carried out in conjunction with process mining, i.e., we do not assume some a priori model.

In [18] CPNs have been used to develop a prototype computer tool for task scheduling. A CPN model of the particular planning domain was created, and a simulation image of this model was extracted and directly used in the tool to calculate schedules (based on state space exploration algorithms), thereby automatically bridging the gap between the formal specification and its implementation. While a domain-specific graphical user interface (GUI) enables planners to modify parameters (i.e., the initial state of the CPN model), the structure of the CPN remains unchanged (and hidden to the planner). With the `CPN Export` plug-in presented in this paper, we support the generation of CPN models for arbitrary business processes in a variety of configurations, which results in very differently structured models.

In [11] a translation of Protos simulation models to CPN Tools is presented. 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 [11], 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.

This paper is based on a paper for the CPN workshop [26]. (The best papers of this workshop were selected for this special section.)

3 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).

The example process used throughout the paper outlines the processing of a liability claim within an insurance company: first, some data related to the claim is registered (cf. activity A in Figure 2), 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. To generate a simulation model that reflects as close as possible the process

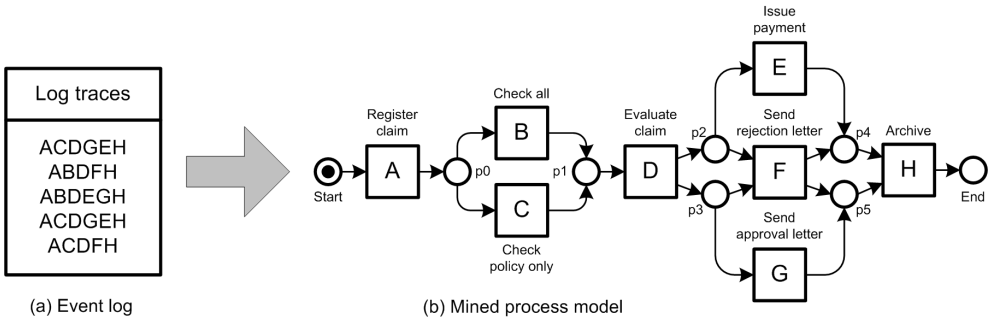


Fig. 2. Process mining phase: based on a set of log traces a process model is constructed

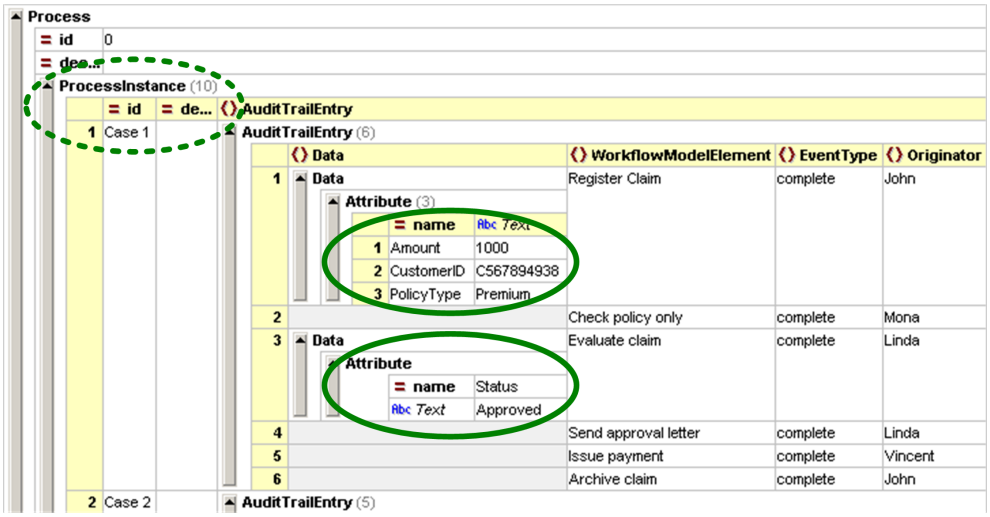


Fig. 3. Fragment of the whole example log in MXML format viewed using XML Spy

that has been observed, data attributes, time stamps, 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. A process log in MXML contains several *process instances* (i.e., cases), whereas each process instance contains a number of *audit trail entries* (i.e., events). The depicted screenshot shows the details about one of the process instances (cf. dotted oval in Figure 3), which contains six audit trail entries. In the following we will have a closer look at which information can be found in the log, considering the perspectives mentioned.

(a) *Data perspective.* Here a data attribute within an audit trail entry (i.e., an event) is interpreted as a case data attribute that has been created, or modified. In the example log one can observe that only activities *Register claim* and *Evaluate claim* have associated data attributes (cf. the two bold ovals in Figure 3). During the execution of activity *Register claim* information about the amount of money 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, the *Amount* attribute 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 may record *schedule*, *start*, and *complete* events (including time stamps) for each activity. Thus, time information can be used to infer, e.g., activity durations, or the arrival rate of new 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 in total 10 process instances, of which 7 executed activity *Check policy only* and only 3 performed the full check procedure *Check all*.

[2] Both the corresponding schema definition and the ProM*import* framework [13], which converts logs from a wide variety of systems to the XML format used by ProM, can be downloaded from *www.processmining.org*.

(c) *Organizational perspective.* In Figure 3 one can observe an event carrying information about the resource that executed the activity. Often, resources are people. However, in principle a resource can be anything (for example, an application performing automated tasks in the process). In the whole insurance claim handling example process (i.e., considering all the 10 cases), 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 to discover data dependencies that influence the routing of a case. The idea of decision mining is briefly explained in the next section (see [23] for further details), and implemented as a plug-in in ProM. The Decision Miner constructs an enhanced model incorporating the data perspective (highlighted by the depicted *Decision Rule* in Figure 1) and passes this on to the *CPN Export*. However, in addition to the control-flow and data perspective the enhanced model may also contain information about probabilities, time, and resources (i.e., the performance and organizational perspectives). The representation of all these perspectives in terms of a generic CPN model, and the capabilities of the `CPN Export` plug-in in ProM, are described in Section 5 and Section 6.

4 Decision Mining

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 with the cases in the event log we subsequently want to find rules for following one route or the other [24].

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 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* (seen from this place, either E or F can be executed) and *p3* (seen from this place, either F or G may be carried out). The idea is to convert every decision point into a *classification problem* [20,21,29], 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 data attributes contained in the log, and we assume that all attributes that have been

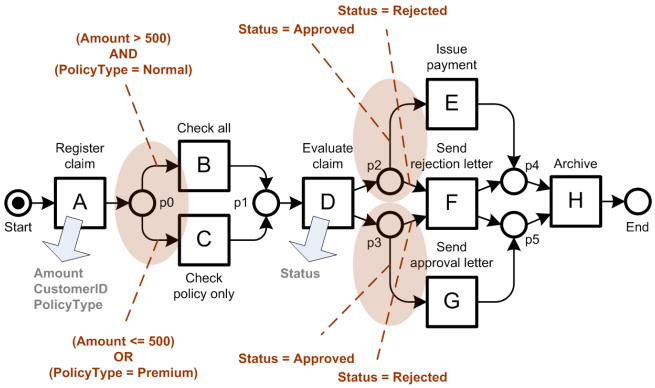


Fig. 4. Enhanced process model

written *before* the choice construct under consideration are relevant for the routing of a case at that point[3].

However, because there is no explicit information in the log about which decision was made at a decision point for some process instance, we first have to infer this information from the log. Starting from the identification of a choice in the process model (i.e., a decision point) 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 to classify the possible decisions. However, 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 [23], where these issues are addressed in detail.

After identifying a decision point in a business process and classifying the decisions of all the process instances in the log, the next step is to determine whether decisions might be influenced by case data, i.e., whether cases with certain properties typically follow a specific route. To solve the formulated classification problem, various algorithms are available [20,29]. We decided to use an algorithm based on decision trees (the C4.5 algorithm [21] to be precise). 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 *overfitting* the data (i.e., that the tree is too much tailored towards the particular training examples).

[3] 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.

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).

Now that we have automatically discovered a model integrating both 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 5), and show how such a CPN model can be generated in ProM (Section 6). Recall that we have also developed mining techniques for discovering these additional perspectives [3,15]. However, a detailed description is beyond the scope of this paper.

5 CPN Model of a Business Process

The goal is to define a CPN representation that can be used to capture process characteristics as, e.g., obtained via process mining techniques. To ensure practical applicability, the chosen CPN model must be generic and suitable for automatic generation. At the same time it must be readable to a human analyst to enable further evaluation and manipulation of the model. Here, the hierarchy concept helps to create understandable components, and separate different layers of abstraction.

In the following we illustrate different perspectives in isolation with the help of the insurance claim handling example. However, these perspectives can be combined and complement each other (for example, the model may contain both data and time information). In our `CPN Export` plug-in in ProM, the user can select options and/or provide information that will affect the generation of the CPN. Section 6 describes these options in more detail.

5.1 General Structure

Since we want to make use of the simulation facilities of CPN Tools, we provide the actual process model together with a simulation environment (amongst others, to generate arriving cases and to measure performance indicators). The top-level page in the hierarchical CPN model is shown in Figure 5(a). For each process model this page will look identical; the environment generates cases and puts them into the *Start* place. Finally, it removes those that have 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*.

Figure 5(b) depicts the simulation environment in more detail. One can observe that the CASE_ID color set is used to refer to particular process instances (i.e., cases). To give each case a unique ID a counter is simply incremented for each generated process instance. For the data perspective, a separate token containing the case ID and a record of case data attributes (defined via the DATA color set) is created and initialized. The initial values represent default values for data attributes until they are explicitly specified. For example, it is desirable to set the *Status* attribute initially to "Rejected" so that—if for some reason the evaluation activity was skipped—a claim could not be automatically approved. Note that the initial values cannot be automatically discovered from our example log, but they can be specified in the `CPN Export` in ProM (cf. Section 6)[4]. The place *Case data* is modeled as a fusion place as activities may need to inspect or modify data attribute values on different pages 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 control the generation of new cases. In Figure 5(b) a constant time delay of 3 is used to realize an arrival process where 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 (e.g., a negative exponential delay to realize a poisson arrival process).

Figure 5(c) 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.

Every activity on the process page has its own sub-page containing the actual simulation information. Depending on the covered perspectives (and their configuration) these activity sub-pages may look very different. In the remainder of this section we will present how certain process characteristics can be represented in terms of a CPN sub-page.

[4] However, if the initial values would be somehow visible in the event log, we could convert this into an initialization event and thus discover the initial probability distribution.

[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 to avoid side effects on other tokens that should actually not be delayed (such as the *Case data* token).

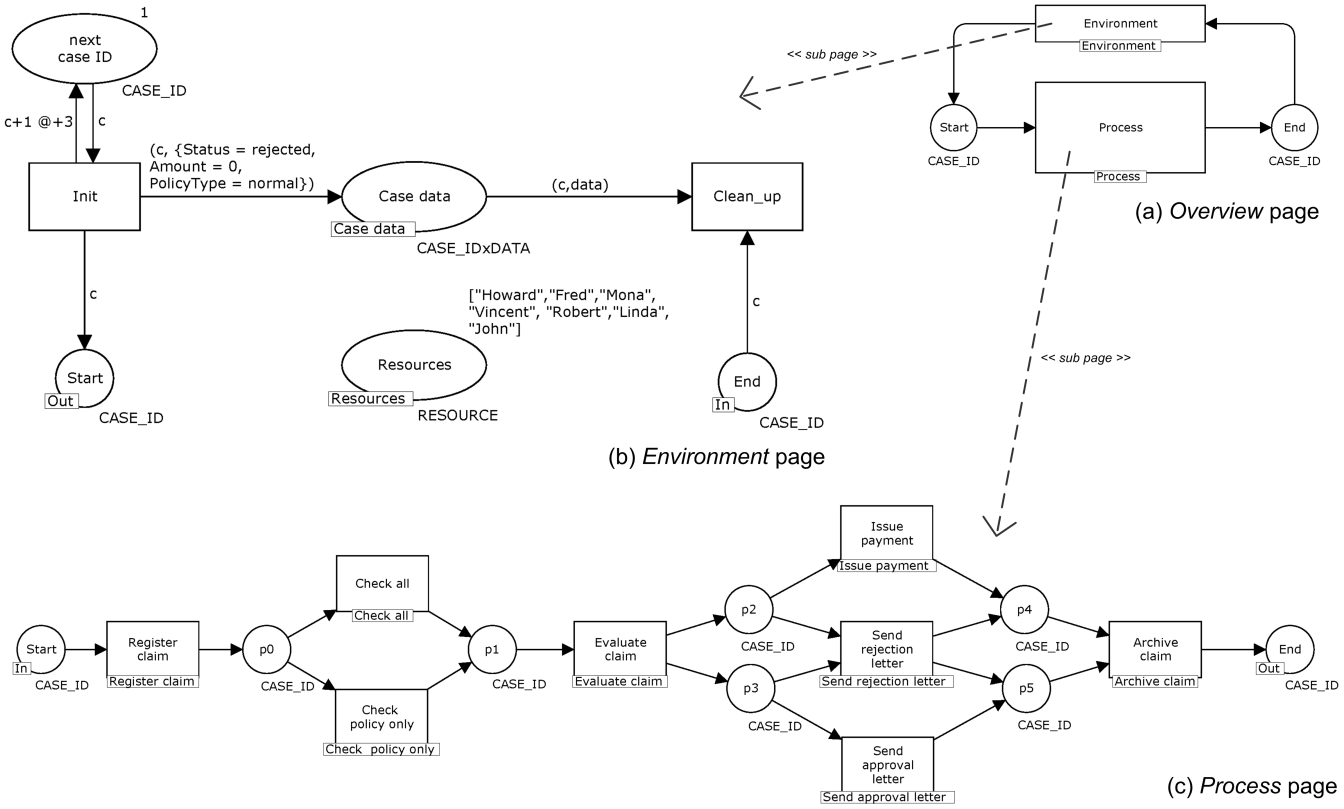


Fig. 5. Hierarchical structure of CPN model

5.2 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 (see the result described in Section 4). 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).

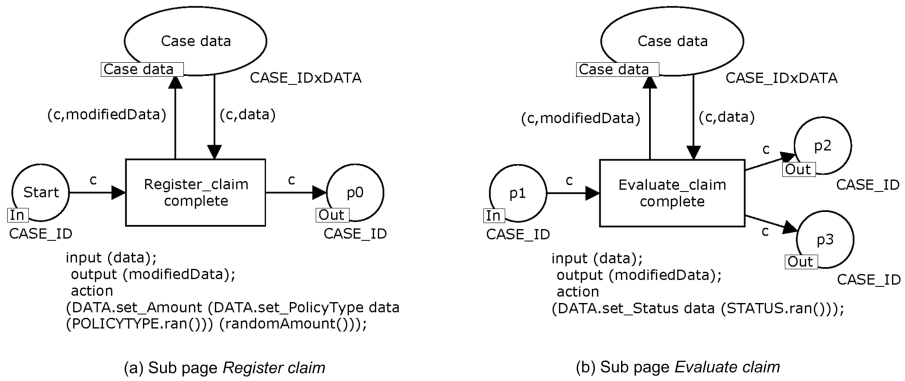
Figure 6 shows how the provision of case data can be simulated using random values. While a random value for a nominal attribute can be generated by applying the *ran()* function directly to the color set[6], a dedicated random function is needed to simulate numeric attributes. In the action part of transition *Register_claim complete* the function *POLICYTYPE.ran()* is used to randomly select the *Policy type* ("Normal" or "Premium"). Furthermore, a function *randomAmount()* was declared to generate a random *Amount*. In this case, the amount is sampled from a discrete distribution generating a value between the lowest and the highest attribute value observed in the event log. However, many other settings are possible. At the end the modified data values are stored in the corresponding *Case data* token (manipulating the corresponding entries using the *set* functions on the DATA record color set).

Figure 7 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 to be fired. Note that the sub-page of activity "Archive claim" is not depicted here as it neither provides nor depends on a data attribute.

5.3 Time

Although there is no time information in the example event log, we want to include the time dimension in our simulation model because it is relatively easy to extract from most real-life logs. Moreover, this perspective is of utmost importance from a practical point of view. To explain this perspective we assume 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 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 7(a) models the activity *Check all* in an atomic

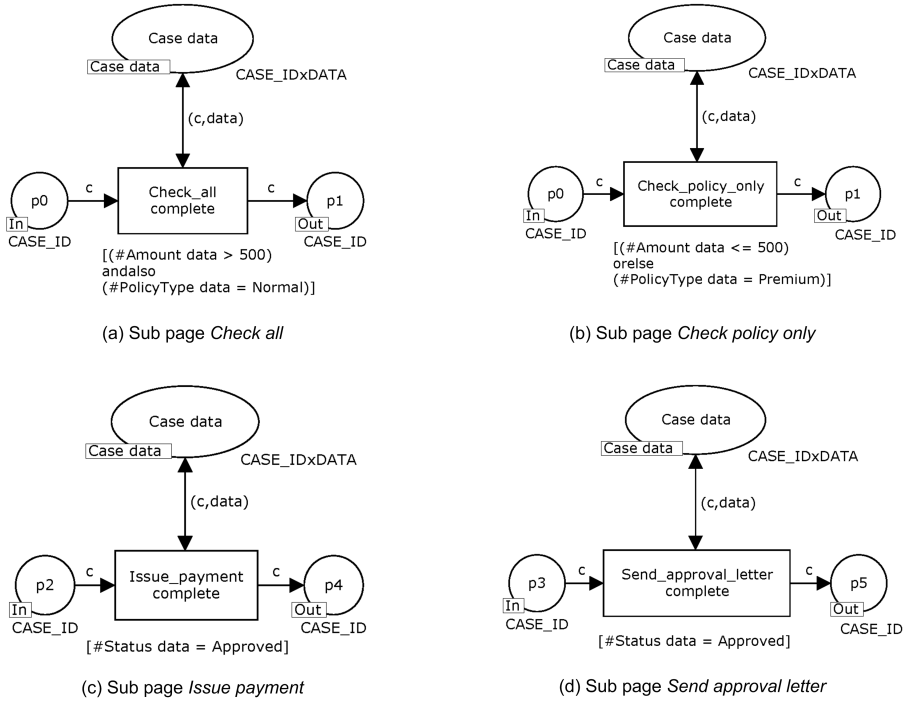
[6] Note that—for performance reasons—the *ran()* function can only be used for enumerated color sets with less than 100 elements.



(a) Sub page *Register claim*

(b) Sub page *Evaluate claim*

Fig. 6. Writing data attributes using random values



(c) Sub page *Issue payment*

(d) Sub page *Send approval letter*

(e) Sub page *Send rejection letter*

Fig. 7. Modeling data dependencies using transition guards

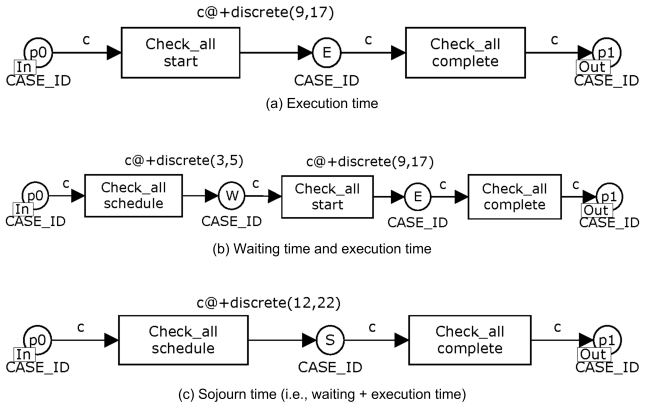


Fig. 8. Different variants of modeling time on sub-page *Check all* depending on the event types (i.e., schedule, start, and complete) present in the log

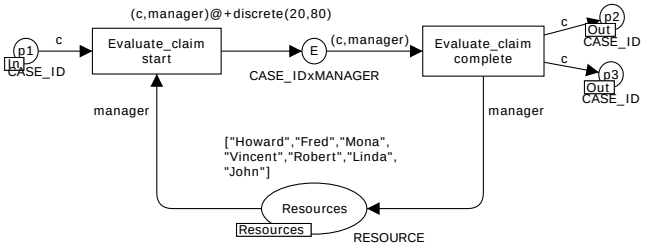


Fig. 9. Sub-page *Evaluate claim* including resource modeling

way, one can distinguish between *schedule*, *start*, and *complete* transitions to incorporate the *waiting time* and *execution time* of this activity. Figure 8 shows three ways to model this for activity *Check all*.

In Figure 8(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 5(b), the token will remain between 9 and 17 time units in place E (i.e., the activity is in the state *Executing*) before transition *Check_all complete* will fire.

In Figure 8(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* will fire.

In Figure 8(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. (This is for example the case for the event log of the Staffware system.)

5.4 Resources

To gain insight into the organizational perspective we can, for example, analyze the event log with the *Social Network miner* of ProM [3]. One possible analysis is to find resources that perform *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. For example, 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 rather similar and quite different from the other three people; they are the only people performing the *Evaluate claim* activity, although they also execute other activities (such as *Send rejection letter* and *Archive claim*). One explanation could be that the activity *Evaluate claim* requires some *Manager* role, whereas all the remaining activities can be performed by people having a *Clerk* role.

A simple way to incorporate this information in our simulation model is to create three groups of resources, namely Finance = {Howard, Vincent}, Manager = {Fred, Linda}, and 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). As indicated, this resource classification can be discovered semi-automatically[7]. However, it could also be derived from some explicit organizational model. Figure 9 depicts how the fact that activity *Evaluate claim* requires the role *Manager* is modeled in the corresponding CPN model. The role is modeled as a separate color set MANAGER, which contains only "Linda" and "Fred". Because the variable *manager* is of type MANAGER, only the resource token "Linda" 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.

5.5 Probabilities and Frequencies

Closely related to the modeling of time aspects is the likelihood of taking a specific path. Both may be of a stochastic nature, i.e., a time duration may be sampled from some probability distribution, and similarly, the selection of an alternative branch may be selected randomly (if there are no data attributes clearly influencing the choice). Hence, the probabilistic selection of a path also needs to be incorporated in the CPN model. Figure 10 shows how often each arc in the model has been used, determined through the log replay analysis carried

[7] Note that the *name* of a group cannot be automatically discovered from the log, but it can be specified in the `CPN Export` in ProM (cf. Section 6).

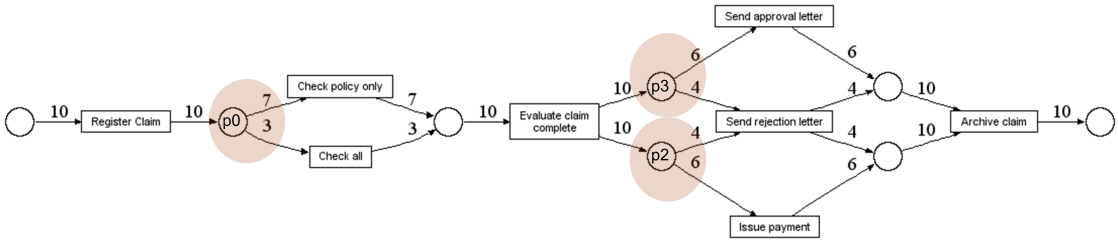


Fig. 10. Frequencies of alternative paths in the example model

out by the *Conformance Checker* in ProM[8]. 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.

To reflect 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 a single place and multiple output transitions. We can 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 11 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 a random value between 0 and 99. After each firing of either transition *Check_all start* or transition *Check_policy_only start* a new random value between 0 and 99 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 to be enabled, which will only happen in 30% of the cases.

Dependent choices The second choice construct in the example model actually consists of two *dependent choices*. This means that the choices represented by places *p2* and *p3*) cannot be considered in isolation; they need to be coordinated to consistently either approve or reject a claim. Therefore, it is clear that two dependent choices cannot be controlled properly by two independently generated probability tokens, because the CPN model will deadlock as soon as the values of the probability tokens indicate contrasting

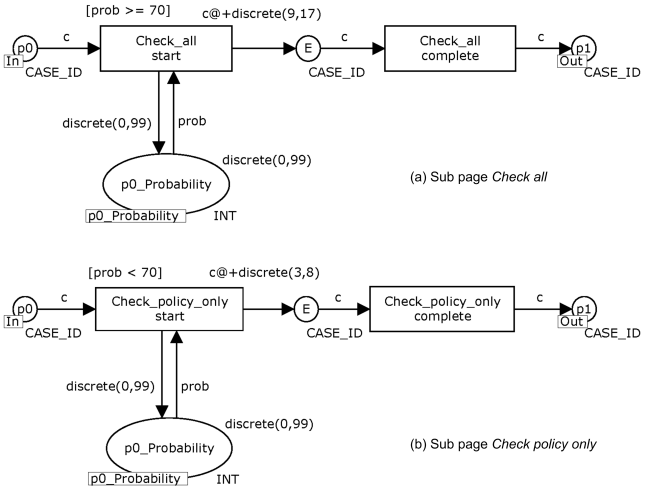


Fig. 11. Using a probability token for simple choices

decisions (e.g., the probability token in *p2* indicates a reject while the other probability token in *p3* suggests to approve the claim).

Figure 12 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 (using the fact that during simulation in CPN Tools all enabled transitions will be fired with an equal probability). The activity duplication is realized on an intermediate sub-page (between process and activity page), which points to multiple instances of the actual activity sub-page (i.e., the activity sub-page is only modeled once). This way, the observed relative frequency[9] of the transitions involved in the dependent choices can be incorporated in the simulation model (i.e., the more likely an activity is, the more instances of that activity are contained on its intermediate sub-page). Figure 12(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*. Figures 12(b) and (c) show similar intermediate sub-pages for the activities *Send approval letter* (also

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

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

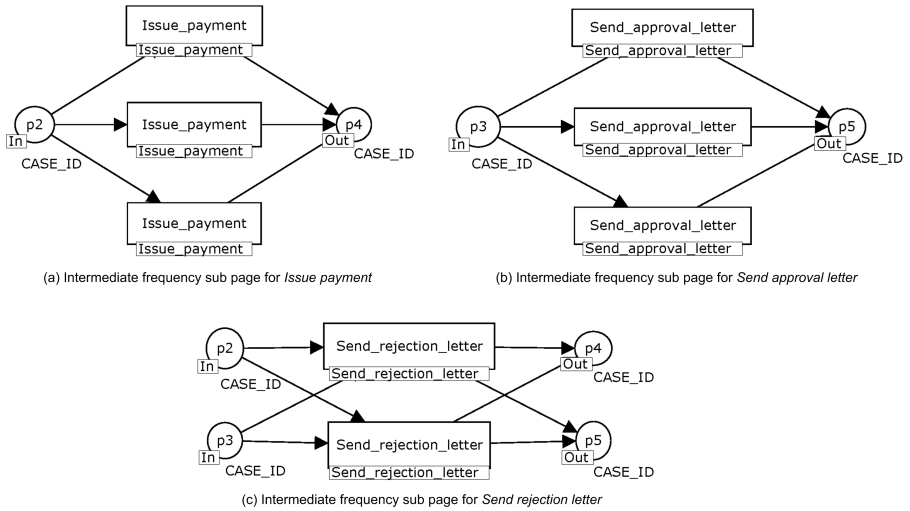


Fig. 12. Modeling dependent choices via activity duplication

duplicated three times) and *Send rejection letter* (duplicated twice).

More advanced (and better scalable) solutions may seek to detect dependencies between choices, and coordinate them, e.g., via a shared probability token.

5.6 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 (a) the utilization of resources and (b) the throughput times of cases during the simulation run. This information can be automatically obtained (see also Section 7) via *data collector monitors* as described in the following.

(a) *Resource utilization.* If resources have been specified 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 define a *marking size monitor* [10] 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.

(b) *Throughput time.* If the time perspective is covered, tokens are created with a time stamp. We record the time stamp of each case's creation together with the case ID token that is routed through the process. This way, we can determine the throughput time of a case by defining a *data collector monitor* [10] for the *Clean up* transition on the *Environment* page (cf. Figure 5(b)), 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.

Note that these are only two examples of possible measures that can be interesting. For example, the current run time of a case could be easily determined at any stage in the process via adding some custom monitor. In Section 7, we briefly demonstrate, how such monitoring components can be used for simulation-based performance analysis of the given model.

Moreover, we want to generate process execution logs for the business process in the CPN 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.

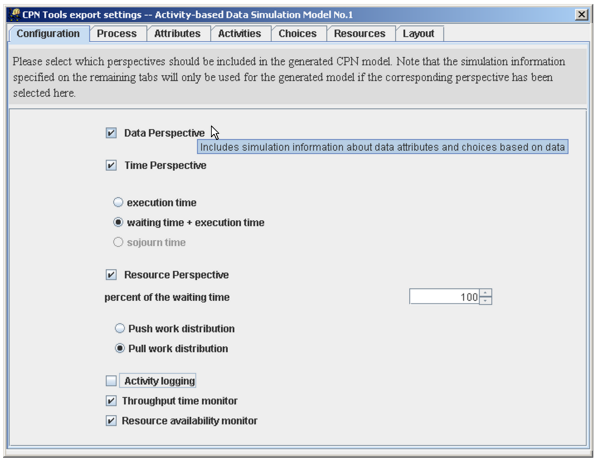
For each firing of a transition on an activity sub-page an event is logged, which includes case ID, the type of transition (i.e., *schedule*, *start*, or *complete*), current time stamp, originator, and additional data (if available). For generating these process execution logs we use the logging functions that have been described in [19]. However, in contrast to [19]—where the code segments of transitions have to be modified to invoke these logging functions—we decided to use *user defined monitors* [10] to clearly separate the logging from the rest of the simulation model.

6 Exporting CPN Models from ProM

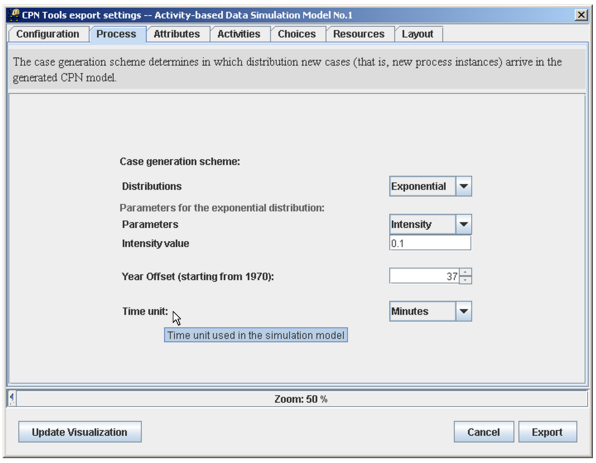
We are able to generate CPN models as presented in the previous section (i.e., including simulation environment and the described monitors) using the `CPN Tools 2.0 Export` plug-in in the ProM framework[11]. It accepts a

[10] Because the type of the current model time is infinite integer and 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.

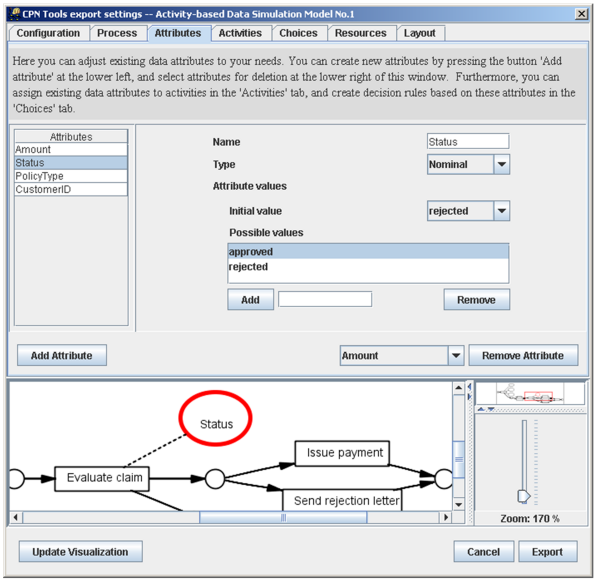
[11] Note that the layout of the generated models was slightly adjusted to improve the readability.



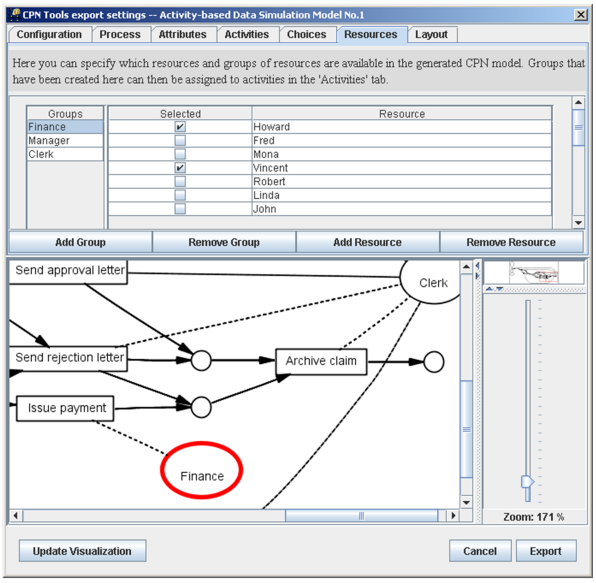
(a) Configuration settings. Determine which perspectives need to be included in the generated simulation model, and how they are to be combined



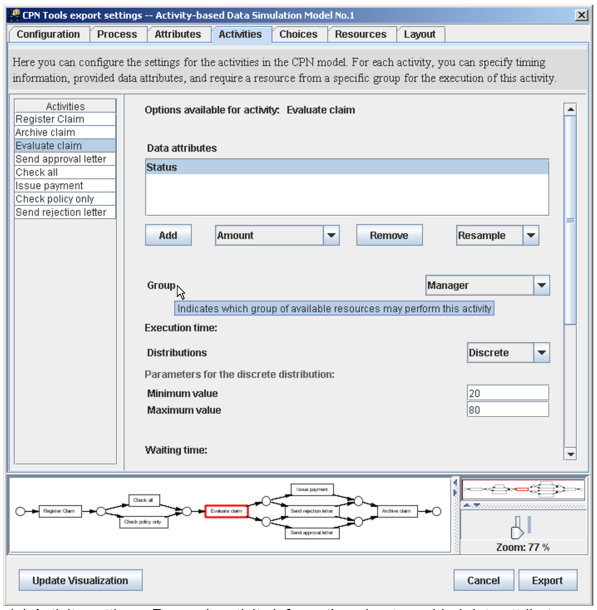
(b) Process settings. Specify how often new cases arrive in the simulation model, and what the time unit and the year of reference are for the MXML logging



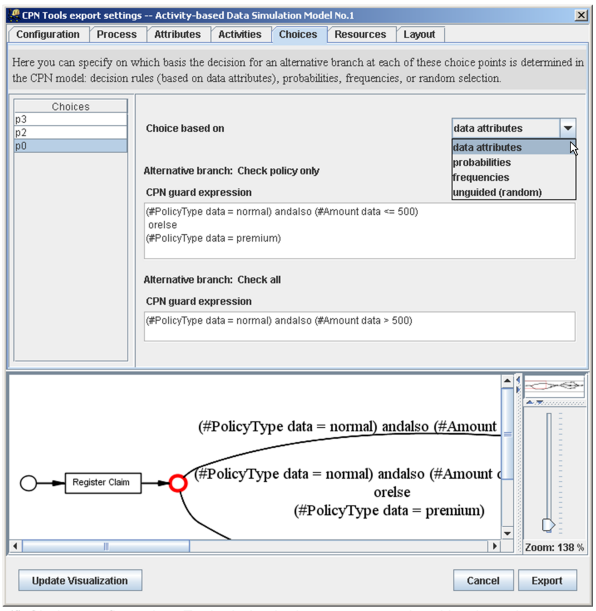
(c) Data attribute settings. Data attributes can be nominal (i.e., have a list of possible enumeration values) or numeric (specified via some random distribution)



(d) Resource settings. Resources can belong to multiple groups (or roles), which allows to consider assignment schemes for the activities in the simulation model



(e) Activity settings. For each activity information about provided data attributes, the required resource group, and execution and waiting times can be specified



(f) Choice configuration. Each choice in the process can be either based on data attributes (i.e., decision rules), probabilities, frequencies, or completely random

Fig. 13. CPN Tools export settings

simulation model that has been discovered by another plug-in in ProM (for example, the enhanced model including data dependencies obtained from the `Decision Miner` plug-in). Alternatively, a simple low-level Petri net can be provided (in this case all information must be provided manually). Before the actual export takes place, the `CPN Export` plug-in allows for the manipulation and configuration of the simulation information in the model. The following options are available:

Figure 13(a) shows the *Configuration settings*, where the user can choose which dimensions should be included in the generated CPN model. In fact, although the relevant simulation information may be provided, it will be ignored if the corresponding configuration option was not chosen. This way, it is easy to play with different configurations of the same simulation model. Note that because the organizational part of the model is rather simple (for example, in reality people may work on multiple processes at the same time, work only 4 days per week, etc.), resources tend to be too "eager" to approximate realistic waiting times for a process. Therefore, it is possible to combine the race for resources among concurrent activities with some explicit waiting time (it can be configured by the user which percentage of the activity waiting time should be used for this). Furthermore, one can choose whether resources should be assigned in a "push" or "pull" mode. In the "push" mode, an activity is already assigned to one specific resource in the scheduling phase, whereas in the "pull" mode any available resource that has the required role can start executing the activity. The monitors described in Section 5.6 are automatically generated if the *Activity Logging* (i.e., MXML logging), *Throughput time monitor*, or *Resource availability monitor* option is selected, respectively.

In Figure 13(b) one can see the *Process Settings*, where the user can adjust global parameters such as the case arrival rate, or the interpretation of one time unit in the CPN model (which is relevant, e.g., for the MXML logging). The case arrival rate can be automatically discovered by the `Performance Analysis with Petri net` plug-in in ProM.

Figure 13(c) depicts the *Attribute 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 probability distribution[12] for a numeric one), and initial value. Note that for our example process the available data attributes were already discovered by the `Decision Miner` plug-in. We can now choose an appropriate initial value for each of them. For this particular example it makes sense to delete the *CustomerID* attribute, since it is not involved in any of the discovered data dependencies.

[12] The `CPN Export` plug-in supports all probability distributions currently available in CPN Tools that are meaningful for the specification of execution times etc., namely constant, binomial, discrete, erlang, exponential, normal, and uniform distribution.

In Figure 13(d) the *Resource settings* are depicted. Here, one can add groups and resources, and assign resources to groups. This way, the `CPN Export` plug-in also supports the specification of information about resources. This information is then used when creating the sub-pages shown earlier. Note that the `Organizational Miner` in ProM can be used to discover groups of resources, which are then called "group1", "group2" etc. In the *Resource settings*, a meaningful name for these discovered groups can be provided before the actual CPN model is generated.

In Figure 13(e), a screenshot of the *Activity settings* for activity *Evaluate 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 attached data attributes were already provided by the `Decision Miner` plug-in (recall that if further information would have been discovered from the log, it would be "filled in" as well), and we can decide whether the old value of the attribute should be kept (i.e., reuse) or whether a random value will be generated (i.e., re-sample). Furthermore, we can assign some execution time (note the discrete distribution between 20 and 80 time units), and choose the suitable group of resources from the list of groups available in the process (note the *Manager* role).

Figure 13(f) 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 5.5), 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 13(f) the data dependency settings are displayed for the choice point *p0*. We can see the data-based decision rules which were discovered and filled in by the Decision Miner. In the current version of the export plug-in such a dependency value is simply a string containing the condition to be placed in the transition guard of the corresponding CPN transition. Alternatively, a probability may be provided between 0.0 and 1.0 for every alternative branch. As discussed in Section 5.5, for dependent choices one should specify a relative frequency value instead.

To demonstrate what can be done with a generated CPN model, the following section highlights the simulation capabilities offered by CPN Tools.

7 Simulation in CPN Tools

Although it is possible to do a state space analysis of the model using CPN Tools, this is not tractable in most practical settings. Therefore, we exploit the fact that CPN Tools also allows for performance analysis based on simulation [17]. Monitors can be used to collect data in log files, create simulation reports, and even generate

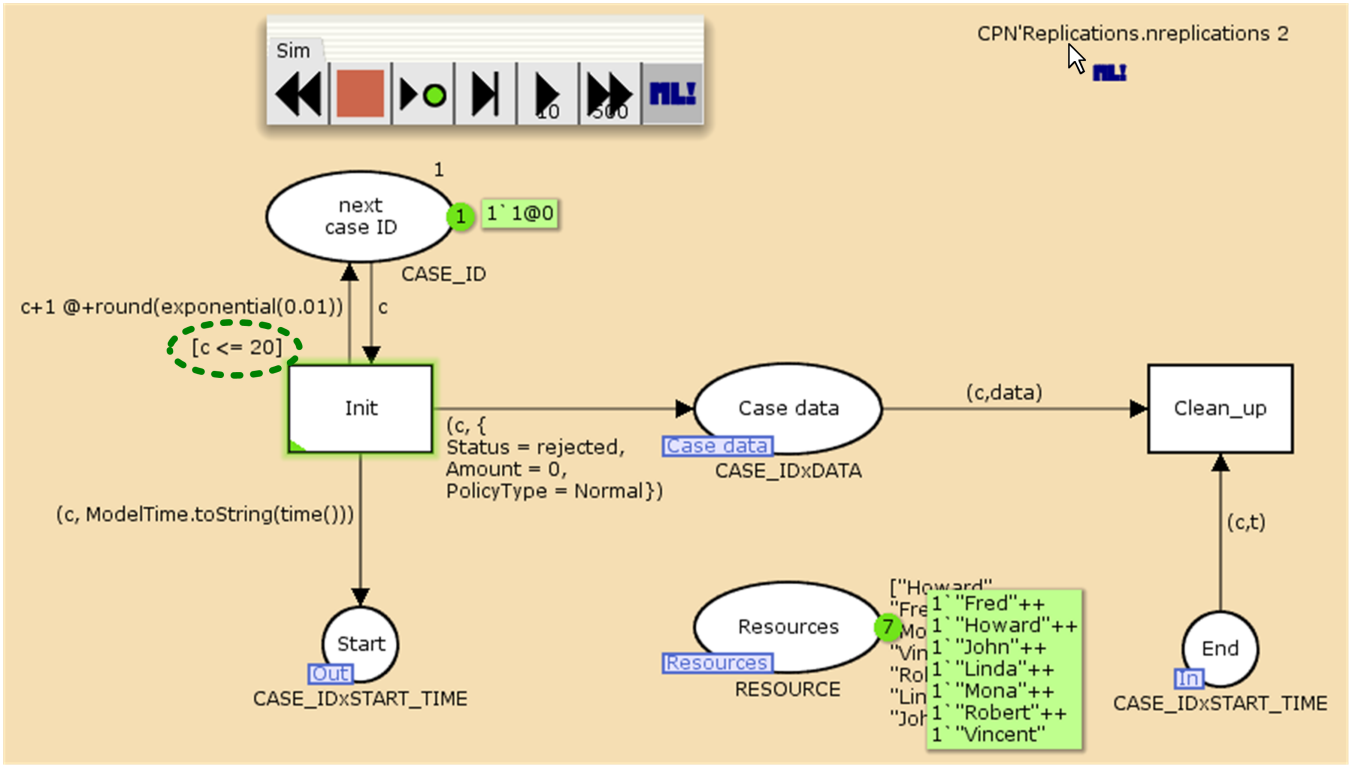


Fig. 14. Running automatic simulation replications in CPN Tools using the ML function `CPN'Replications.nreplications` n

Gnuplot scripts that visualize the collected data from different simulation runs.

We take a CPN model that was automatically generated by the `CPN Export` plug-in in ProM as the starting point. The model already contains a simulation environment that generates new cases according to a certain distribution. Let us assume that the case generation scheme of our insurance claim handling example process follows an exponential distribution with the intensity value 0.01. This corresponds to a mean inter-arrival time of 100 time units (which we here interpret as minutes). Furthermore, for simplicity, we assume that each activity in the process has a constant waiting and execution time of 30 minutes; only activity *Check all* has a (constant) execution time of 60 minutes. The generated CPN model already contains two monitors for measuring resource utilization and throughput time as described in Section 5.6. However, it is easy to add further monitoring components. For example, we additionally want to measure the number of cases in the process (i.e., the current work load). For this, we simply select the pre-defined *marking size monitor* tool and apply it to the *Case data* place, which holds one token for each case that is currently handled by the process.

The model can be readily simulated in CPN Tools for an arbitrary number of steps. In the meantime, numerical data about the resources utilization, throughput times, and work load is automatically extracted from the model and recorded in log files. From this, a performance report indicating statistical measures such as minimum, maximum, and average values is generated, and confidence intervals can be used to indicate how precise these estimates of a performance measure are. However, for statistical validity, independent and identically distributed (iid) estimates of the performance measures must be collected. It is clear that, for example the throughput time of a case is influenced by other cases in the process as they fight for the same resources. Therefore, measures collected from a single simulation run are *not* independent. CPN Tools provides support for performance analysis using simulation replications of independent, terminating simulations, and Figure 14 shows how such simulation replications can be automatically run using an auxiliary text field containing the ML function `CPN'Replications.nreplications` n, whereas n determines the number of replications (here $n = 2$). Note that to let each simulation run terminate, we limited the number of cases to be generated (cf. dotted oval in Figure 14).

If we run two such replicated simulations, with each 20 insurance claims being handled by the process, CPN Tools automatically generates Gnuplot scripts that can be used to visualize the data collected by the three monitors. For example, the two top-most graphs in Figure 15 visualize the throughput times of finishing cases and the number of cases in the process over time for the two simulation runs. One can see that in the beginning of the second simulation run there were up to 10 cases pro-

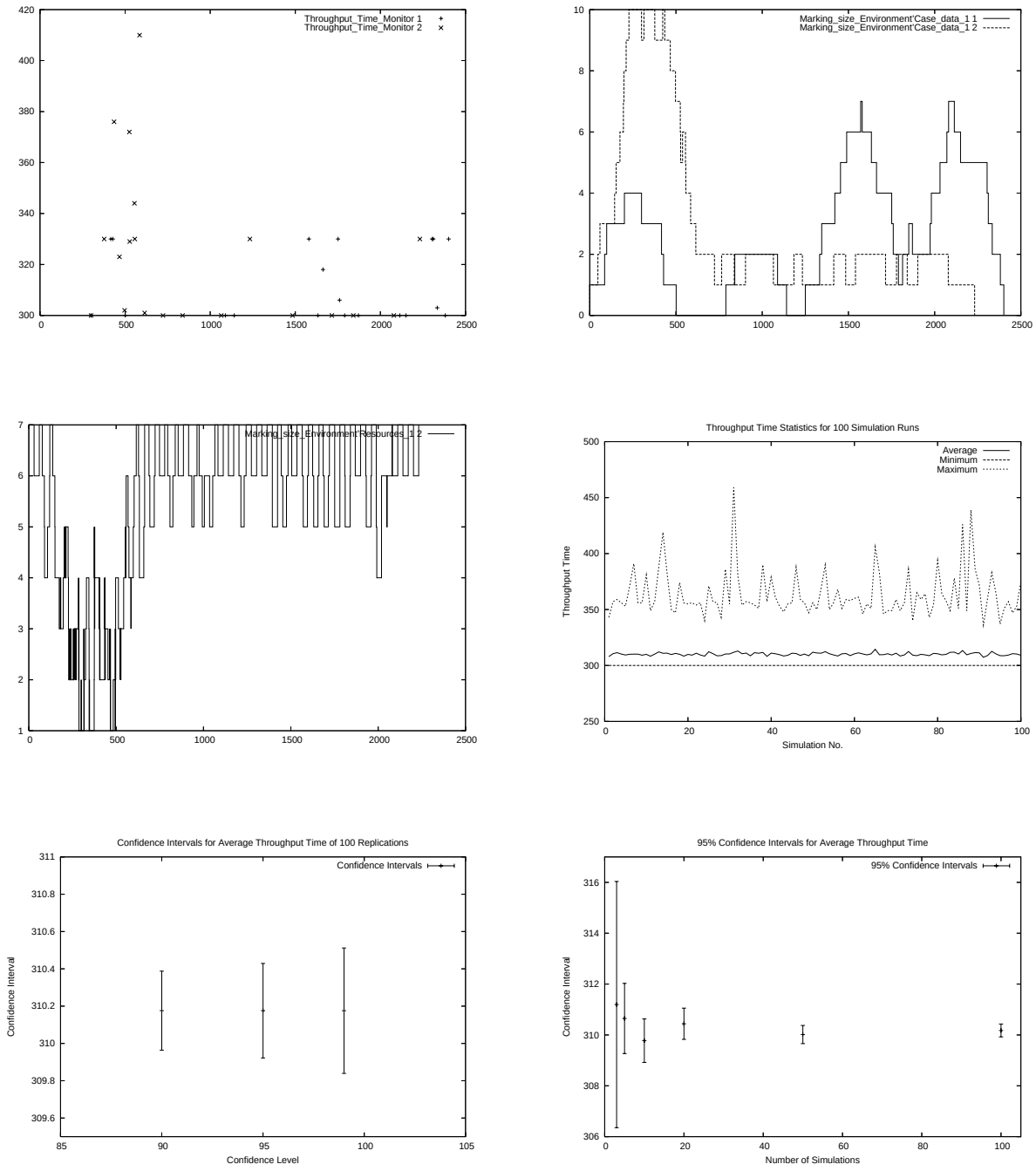


Fig. 15. Gnuplot graphs that are plotted based on log files written by data collector monitors in CPN Tools

cessed at the same time. Furthermore, the throughput times of finishing cases were considerably higher in the first phase of the second simulation, which hints that cases were delayed because of the unavailability of resources. This can also be observed in the middle-left graph in Figure 15, which displays the number of resources that were available over time for the second simulation run only.

While this gives an idea of the developments within a single simulation run, we are also interested in general statistics, e.g., about the throughput times of cases. For this, we need a larger data basis and, therefore, we run 100 simulations with each 200 cases being handled by the process. Figure 16(a) shows a screenshot of the performance output options in CPN Tools, where one can select the measures to be calculated. After the sim-

(a) Options for performance output in CPN Tools

90%, 95%, and 99% Confidence intervals for average Throughput Time

(see also graph on bottom-left in Figure 15)

Statistics

Name	Avrg	90% Half Length	95% Half Length	99% Half Length	StD	Min	Max
Marking_size_Environment'Case_data_1							
count_iid	1301.590000	1.180206	1.411994	1.873444	7.088324	1285	1318
max_iid	9.280000	0.206418	0.246957	0.327665	1.239746	7	13
min_iid	0.000000	0.000000	0.000000	0.000000	0.000000	0	0
avrg_iid	3.095284	0.039084	0.046760	0.062042	0.234740	2.611308	3.927983
Marking_size_Environment'Resources_1							
count_iid	2201.180000	2.360412	2.823988	3.746888	14.176649	2168	2234
max_iid	7.000000	0.000000	0.000000	0.000000	0.000000	7	7
min_iid	0.080000	0.045398	0.054314	0.072064	0.272660	0	1
avrg_iid	5.275778	0.021450	0.025662	0.034049	0.128827	4.820618	5.536400
Throughput_Time_Monitor							
count_iid	200.000000	0.000000	0.000000	0.000000	0.000000	200	200
max_iid	363.520000	3.441039	4.116847	5.462262	20.666901	336	459
min_iid	300.000000	0.000000	0.000000	0.000000	0.000000	300	300
sum_iid	62035.120000	42.358353	50.677381	67.239116	254.404525	61468	62889
avrg_iid	310.175600	0.211792	0.253387	0.336196	1.272023	307.340000	314.445000

(b) Replication performance report based on 100 simulation runs of the insurance claim handling example, with each 200 cases being handled by the process. Three monitors collected data on workload, resource availability, and throughput time

Fig. 16. Performance reports are automatically generated by CPN Tools

ulation has finished, a replication performance report as depicted in Figure 16(b) is automatically generated by CPN Tools. For example, from the data collected by the throughput time monitor, the average minimum (*min_iid*), maximum (*max_iid*), accumulated (*sum_iid*), and average (*avrg_iid*) values are calculated for the 100 simulation runs. The number of samples taken (*count_iid*) is equal for all the replicated simulations as 200 cases are the stop criteria to finish a simulation run.

In addition to the performance report, which provides an overview of the performance measures from the replicated simulations, CPN Tools also records the statistics for each of the simulation runs in log files. Although for this there is no Gnuplot script generated by CPN Tools, we can easily plot the developments of, e.g., average, minimum and maximum values based on these log files with the help of a custom Gnuplot script as shown in the middle-right graph in Figure 15 (in our custom Gnuplot script we now also added meaningful labels for the axes). One can see that compared to the maximum throughput time, the average throughput time is relatively stable across the 100 simulations. The minimum throughput time is 300 minutes for every single simulation run, which can be explained by the chosen constant waiting and execution time for the activities in the model. If at the end of the process the insurance claim is rejected, five activities are executed for this case. If there was no delay in processing the case (due to the unavailability of resources), and only a policy check was performed, then the throughput time is exactly 300 minutes[13].

The replication performance report reflects the precision of a performance measure over the different simulation runs using confidence intervals. For example, Figure 16 highlights the 90%, 95%, and 99% confidence intervals for the average throughput time, e.g., [309.96, 310.39] is the 90% confidence interval. Similar to the observations from the middle-right graph in Figure 15, one can see that—compared to the average throughput time confidence intervals—the confidence intervals for the maximum throughput time are much bigger, and for the minimum throughput time they have length 0.

In addition to the performance report itself, the confidence intervals for the various performance measures are also recorded in confidence interval files. Again, these data can be used to create custom Gnuplot graphs visualizing, e.g., the 90%, 95%, and 99% confidence intervals for the average throughput time as depicted in the bottom-left of Figure 15. The bottom-right graph in Figure 15 demonstrates the effect of the number of repli-

[13] Note that the constant execution and waiting time was chosen for demonstration purposes. However, to approximate a real-life situation one would rather discover, e.g., a service time distribution from the time information in the event log that was used to discover the simulation model.

Table 1. Effect of the individual modifications on the average throughput times (95% confidence interval)

CPN Model	*Average throughput times*
Initial model	[309.92, 310.43]
(1) Always full check	[333.36, 333.94]
(2) Twice as many resources	[307.72, 308.13]
(3) Increased case load	[3400.25, 3446.39]
(4) Sequentialized tasks	[339.67, 340.53]

cations (here 3, 5, 10, 50, and 100 replications were run during a simulation) on both the estimate of the performance measure as well as the confidence intervals. In the graph of Figure 15 95% confidence intervals for the average throughput time are plotted, and one can see that the estimated performance measures become more accurate with an increasing number of replications.

Simulation-based performance analysis as shown in this section can be used to explore "what if" scenarios. For example, one could predict the flow times for alternative configurations (e.g., less specialists for a certain task) or re-designs of the process under consideration. In the following, we demonstrate the effect of a few simple modifications in the context of the running example: (1) We removed the option to do a partial check (i.e., *Check policy only*), which takes 30 minutes, and always perform the full check (i.e., *Check all*), which takes 60 minutes, (2) We assumed that we have exactly twice as many resources of each type (i.e., there are two "Johns", two "Lindas" etc.), (3) We increased the case load from a mean inter-arrival time of 100 minutes to 10 minutes (i.e., now on average every 10 minutes a new insurance claim arrives at the process), and (4) We sequentialized the tasks *Send approval letter* and *Issue payment* to make sure the letter is sent before the payment is made, as a courtesy to the claimant. Table 1 shows the effect of these individual modifications with respect to the initial model (i.e., as previously described in this section). For every CPN model, we run 100 simulation replications for 200 cases, and give the 95% confidence interval of the average throughput times of the (modified) process. As can be expected, from the four alternative scenarios only the increase of resources leads to decreased throughput times compared to the initial model.

Note that in this section we have treated our insurance claim handling process as a *terminating* process [17], i.e., we have created a number of finite simulation runs to evaluate how the process behaves when handling a certain number of cases. Similarly, one could also evaluate the performance during a certain period of time, for example, a work day at the insurance company (terminating the process after 8 hours of model time). Via simulating terminating systems we seek to study the *transient behavior* of the system. For this, CPN Tools provides support using simulation replications of independent, terminating simulations. In a *non-terminating* system, the system's behavior is not limited to some time window. One could think of the insurance claim handling process also as a non-terminating process handling continuously incoming cases. Such simulations are typically used to investigate the long-term behavior of a system (*steady-state* behavior analysis). See [17] for further information on this topic. Generally, it is important to align the goals of a simulation study with the experimental design. Systematic decisions are required to ensure validity and to produce useful results. However, this is beyond the scope of this paper.

8 Conclusion

In this paper, we have shown that it is possible to discover process models with data from event logs. Furthermore, we have presented a CPN representation that captures a business process from multiple perspectives (i.e., data, performance, and organizational perspective), and which can be automatically generated using the `CPN Export` plug-in in ProM. Finally, we demonstrated how such generated simulation models can be analyzed in CPN Tools.

Future work includes the refinement of the generated CPN models. 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. Moreover, we plan to apply our approach to real-life data. For this, the discovery of further perspectives of a business process will be integrated in the mined process models. Note that existing plug-ins in ProM 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) can then be combined in one comprehensive 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 the Technology Foundation STW, EIT, SUPER, NWO, and the IOP program of the Dutch Ministry of Economic Affairs. Furthermore, the authors would like to thank all ProM developers 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, 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, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713–732, 2007.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. CPN Group, Aarhus, Denmark. CPN Tools Help Pages. http://wiki.daimi.au.dk/cpntools-help/.
11. 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 K. Jensen, editor, *Proceedings of the Seventh Workshop on the Practical Use of Coloured Petri Nets and CPN Tools*, pages 137–156. University of Aarhus, Denmark, 2006.
12. 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.
13. C.W. Günther and W.M.P. van der Aalst. A Generic Import Framework for Process Event Logs. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops, Workshop on Business Process Intelligence (BPI 2006)*, volume 4103 of *Lecture Notes in Computer Science*, pages 81–92. Springer-Verlag, Berlin, 2006.
14. 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.
15. P.T.G. Hornix. Performance Analysis of Business Processes through Process Mining. Master's thesis, Eindhoven University of Technology, Department of Computer Science, Eindhoven, The Netherlands, 2007.
16. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Springer-Verlag, 1997.
17. K. Jensen, L.M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *Software Tools for Technology Transfer (STTT)*, 9(3–4):213–254, 2007.
18. L.M. Kristensen, P. Mechlenborg, L. Zhang, B. Mitchell, and G.E. Gallasch. Modelbased Development of a Course of Action Scheduling Tool. In K. Jensen, editor, *Proceedings of the Seventh Workshop on the Practical Use of Coloured Petri Nets and CPN Tools*, pages 1–16. University of Aarhus, Denmark, 2006.
19. A.K. Alves de Medeiros and C.W. Günther. Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms. In K. Jensen, editor, *Proceedings of the Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 177–190, 2005.
20. T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
21. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
22. 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.
23. A. Rozinat and W.M.P. van der Aalst. Decision Mining in Business Processes. BPM Center Report BPM-06-10, BPMcenter.org, 2006.
24. A. Rozinat and W.M.P. van der Aalst. Decision Mining in ProM. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *BPM 2006*, volume 4102 of *Lecture Notes in Computer Science*, pages 420–425. Springer-Verlag, Berlin, 2006.
25. A. Rozinat, I.S.M. de Jong, C.W. Günther, and W.M.P. van der Aalst. Process Mining of Test Processes: A Case Study. BETA Working Paper Series, WP 220, Eindhoven University of Technology, Eindhoven, 2007.
26. A. Rozinat, R.S. Mans, and W.M.P. van der Aalst. Mining CPN Models: Discovering Process Models With Data from Event Logs. In K. Jensen, editor, *Proceedings of the Seventh Workshop on the Practical Use of Coloured Petri Nets and CPN Tools*, pages 57–76. University of Aarhus, Denmark, 2006.
27. 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.
28. 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.
29. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition*. Morgan Kaufmann, 2005.