Workflow Simulation for Operational Decision Support using YAWL and ProM

A. Rozinat[1], M. T. Wynn[2], W. M. P. van der Aalst[1,2], A. H. M. ter Hofstede[2], and C. J. Fidge[2]

[1] Information Systems Group, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
{a.rozinat,w.m.p.v.d.aalst}@tue.nl

[2] Business Process Management Group, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001, Australia.
{m.wynn,a.terhofstede,c.fidge}@qut.edu.au

Abstract. Simulation is widely used as a tool for analyzing business processes but is mostly focused on examining rather abstract steady-state situations. Such analyses are helpful for the initial design of a business process but are less suitable for operational decision making and continuous improvement. Here we describe a *simulation system for operational decision support* in the context of workflow management. To do this we exploit not only the workflow's *design*, but also logged data describing the system's observed *historic* behavior, and information extracted about the current *state* of the workflow. Making use of actual data capturing the current state and historic information allows our simulations to accurately predict potential near-future behaviors for different scenarios. The approach is supported by a practical toolset which combines and extends the workflow management system YAWL and the process mining framework ProM.
This technical report contains a detailed description of how a simulation model including operational decision support can be generated by our software based on the running example.

Keywords: Workflow Management, Process Mining, Short-term Simulation.

Table of Contents

1 Introduction

Business process simulation is a powerful tool for process analysis and improvement. One of the main challenges is to create simulation models that *accurately* reflect the real-world process of interest. Moreover, we do not want to use simulation just for answering strategic questions but also for tactical and even operational decision making. To achieve this, different sources of simulation-relevant information need to be leveraged. In this paper, we present a new way of creating a simulation model for a business process supported by a workflow management system, in which we integrate design, historic, and state information.

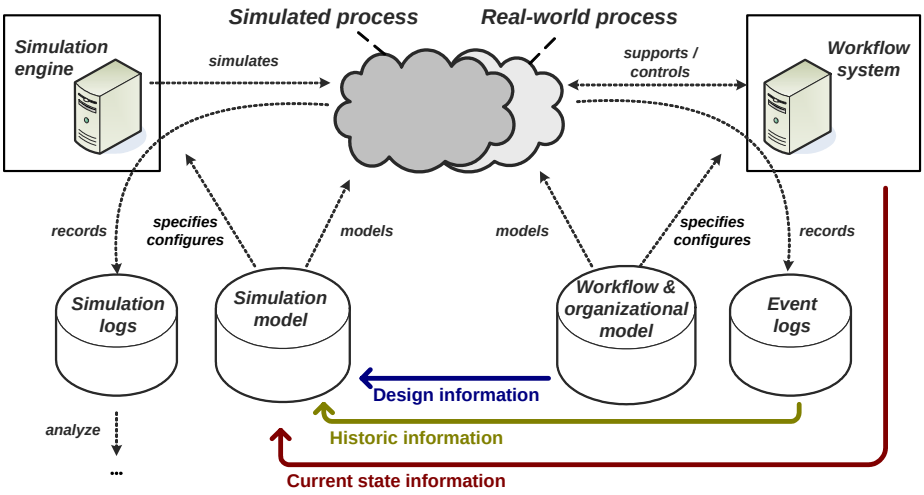


Fig. 1. Overview of our integrated workflow management (right) and simulation (left) system

Figure 1 illustrates our approach. We consider the setting of a *workflow system* that supports some *real-world process* based on a *workflow and organizational model*. Note that the workflow and organizational models have been designed before enactment and are used for the configuration of the workflow system. During the enactment of the process, the performed activities are recorded in *event logs*. An event log records events related to the offering, start, and completion of work items, e.g., an event may be 'Mary completes the approval activity for insurance claim XY160598 at 16.05 on Monday 21-1-2008'.

The right-hand side of Figure 1 is concerned with enactment using a workflow system while the left-hand side focuses on analysis using simulation. In order to link enactment and simulation we propose to use three types of information readily available in workflow systems. These are used to create and initialize the simulation model.

– *Design information.* The workflow system has been configured based on an explicit process model describing control and data flows. Moreover, the workflow system uses organizational data, e.g., information about users, roles, groups, etc.
– *Historic information.* The workflow system records all events that take place in 'event logs' from which the complete history of the process can be reconstructed. By analyzing historic data, probability distributions for workflow events and their timing can be extracted.
– *State information.* At any point in time, the workflow process is in a particular state. The current state of each process instance is known and can be used to initialize the simulation model. Note that this current state information includes the control-flow state (i.e., 'tokens' in the process model), case data, and resource data (e.g., resource availability).

By merging the above information into a simulation model, it is possible to construct an *accurate model based on observed behavior* rather than a manually-constructed model which approximates the workflow's anticipated behavior. Moreover, the current state information supports a 'fast forward' capability, in which simulation can be used to explore different scenarios with respect to their *effect in the near future*. In this way, simulation can be used for *operational decision making*.

Based on this approach, the system design in Figure 1 allows different simulation experiments to be conducted. For the 'as-is' situation, the simulated and real-world processes should overlap as much as possible, i.e., the two process 'clouds' in Figure 1 coincide. For the 'to-be' situation, the observed differences between the simulated and real-world processes can be explored and quantified. In our implementation we ensure that the simulation logs have the same format as the event logs recorded by the workflow system. In this way we can use the *same tools* to analyze both simulated and real-world processes.

To do this, we need state-of-the art *process mining* techniques to analyze the simulation and event logs and to generate the simulation model. To demonstrate the applicability of our approach, we have implemented the system shown in Figure 1 using ProM [1] and YAWL [2]. YAWL is used as the workflow management system and has been extended to provide high-quality design, historic, and state information. The process mining framework ProM has been extended to merge the three types of information into a single simulation model. Moreover, ProM is also used to analyze and compare the logs in various ways.

The paper is organized as follows. Related work is reviewed in Section 2. Section 3 describes the overall approach proposed, presents a running example, and provides a brief overview of the implementation realized using YAWL and ProM. Section 4 then describes on a detailed, step-by-step basis how a simulation model including operational decision support can be generated by our software for the running example. Section 5 concludes the paper by discussing the three main innovations presented in this paper.

2 Related Work

Our work combines aspects of workflow management, simulation, and process mining. Some of the most relevant contributions from these broad areas are reviewed below.

Prominent literature on workflow management [6, 14, 19] focuses on enactment, and research on workflow analysis usually focuses on verification, rather than simulation. Conversely, publications on simulation typically concentrate on statistical aspects [12, 17, 13] or on a specific simulation language [11]. Several authors have used simulation or queuing techniques to address business process redesign questions [4, 5, 15], and most mature workflow management systems provide a simulation component [8, 9]. However, none of these systems uses historic and state information to learn from the past and to enable operational decision making. We are not aware of any toolset that is able to extract the current state from an operational workflow management system and use this as the starting point for transient analysis.

In earlier work we first introduced the notion of using historic and state information to construct and calibrate simulation models [16, 20], and used Protos, ExSpect, and COSA to realize the concept of short-term simulation [16]. However, this research did not produce a practical publicly available implementation and did not use process mining techniques.

Process mining aims at the analysis of event logs [3]. It is typically used to construct a static model that is presented to the end-user to reflect on the process. Previously we showed that process mining can also be used to generate simulation models [18], but design and state information were not used in that work.

3 Approach

In this section, we first describe the overall approach, and presents a running example (Section 3.1). Then, we provide a brief overview of the implementation realized using YAWL and ProM (Section 3.2).

3.1 Overview and Example

A crucial element of the approach in Figure 1 is that the *design*, *historic* and *state* information provided by the workflow system are used as the basis for simulation. Table 1 describes this information in more detail.

The design information is static, i.e., this is the specification of the process and supporting organization that is provided at design time. This information is used to create the structure of the simulation model. The historic and state information are dynamic, i.e., each event adds to the history of the process and changes the current state. Historic information is aggregated and is used to set parameters in the simulation model. For instance, the arrival rate and processing times are derived by aggregating historic data, e.g., the (weighted)

Table 1. Process characteristics and the data sources from which they are obtained

Design information *(obtained from the workflow and organization model used to configure the workflow system)*	**Historic information** *(extracted from event logs containing information on the actual execution of cases)*	**State information** *(based on information about cases currently being enacted using the workflow system)*
• control and data flow (activities and causalities)	• data value range distributions	• progress state of cases (state markers)
• organizational model (roles, resources, etc.)	• execution time distributions	• data values for running cases
• initial data values	• case arrival rate	• busy resources
• roles per task	• availability patterns of resources	• run times for cases

average over the last 100 cases is used to fit a probability distribution. Typically, these simulation parameters are not very sensitive to individual changes. For example, the average processing time typically changes only gradually over a long period. The current state, however, is highly sensitive to change. Individual events directly influence the current state and must be directly incorporated into the initial state of the simulation. Therefore, design information can be treated as static, while historic information evolves gradually, and state information is highly dynamic.

To realize the approach illustrated in Figure 1 we need to merge design, historic and state information into a single simulation model. The design information is used to construct the structure of the simulation model. The historic information is used to set parameters of the model (e.g., fit distributions). The state information is used to initialize the simulation model. Following this, traditional simulation techniques can be used. For example, using a random generator and replication, an arbitrary number of independent simulation experiments can be conducted. Then statistical methods can be employed to estimate different performance indicators and compute confidence intervals for these estimates.

By modifying the simulation model, various 'what-if' scenarios can be investigated. For example, one can add or remove resources, skip activities, remove cases, etc. and see what the effect is. Because the simulation experiments for these scenarios start from the current state of the actual system, they provide a kind of 'fast-forward button' showing what will happen in the near future, to support operational decision making. For instance, based on the predicted system behavior, a manager may decide to hire more personnel or stop accepting new cases.

Importantly, the simulations yield simulation logs in the same format as the event logs. This allows process mining techniques to be used to view the real-world processes and the simulated processes in a unified way. Moreover, both can be compared to highlight deviations, etc.

Consider the credit card application process expressed as a YAWL workflow model in Figure 2. The process starts when an applicant submits an application. Upon receiving an application, a credit clerk checks whether it is complete. If not, the clerk requests additional information and waits until this information is received before proceeding. For a complete application, the clerk performs further checks to validate the applicant's income and credit history. Different checks are performed depending on whether the requested loan is large (e.g. greater than $500) or small. The validated application is then passed on to a manager to decide whether to accept or reject the application. In the case of acceptance, the applicant is notified of the decision and a credit card is produced and delivered to the applicant. For a rejected application, the applicant is notified of the decision and the process ends.

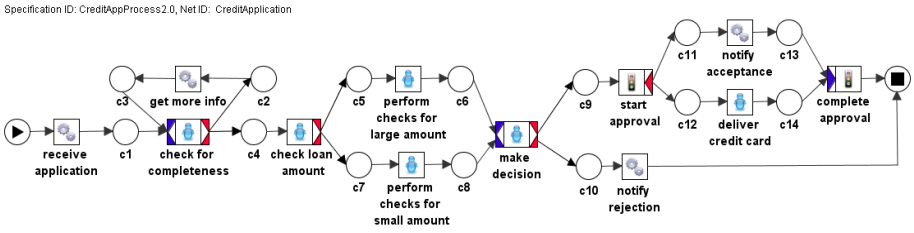


Fig. 2. A credit application process modeled in YAWL

Here we assume that this example workflow has been running for a while. In YAWL but also any other workflow system the following runtime statistics can be gathered about the long-term behavior of this process.

- Case arrival rate: 100 applications per week
- Throughput time: 4 working days on average

With respect to resources, there are eight members of staff available, which include three capable of acting as 'managers' and seven capable of acting as 'clerks'. (One person can have more than one role.)

Further assume that due to a successful Christmas promotion advertised in November, the number of credit card applications per week has temporarily doubled to 200. The promotion period is now over and we expect the rate to decrease to 100 applications per week again. However, as a result of the increased interest, the system now has a backlog of 150 applications in various stages of processing, some of which have been in the system for more than a week. Since it is essential that most applications are processed before the holiday season, which begins in a fortnight from now (the 'time horizon' of interest), management would like to perform simulation experiments from the current state ('fast forward') to determine whether or not the backlog can be cleared in time.

3.2 Realization through YAWL and ProM

We now use the example introduced in Section 3 to describe our proof-of-concept implementation supporting the approach depicted in Figure 1. The realization is based on the YAWL workflow environment [2] and the process mining framework ProM [1]. We focus on the new capabilities that have been added to these systems, and briefly explain the main steps that need to be performed. A detailed description is provided in the tutorial section of this report (cf. Section 4).

Extracting Simulation-Relevant Information The information contained in the workflow specification is supplemented with historical data obtained from the event logs and data from the organizational model database. This was achieved by implementing two new functions in the workflow engine to export historical data from the logs for a particular specification and to export the organizational model (i.e., information about roles and resources).

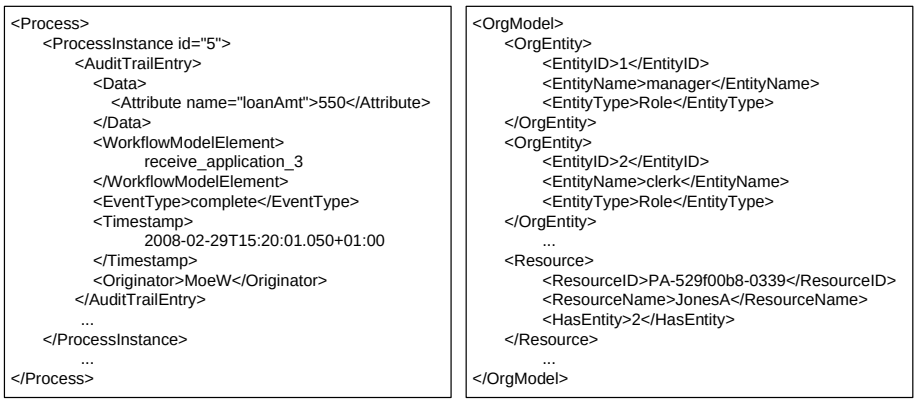
In the YAWL workflow system, event logs are created whenever an activity is enabled, started, completed or cancelled, together with the time when this event occurred and with the actor who was involved. Logs are also kept for data values that have been entered and used throughout the system. Therefore, we can retrieve historical data about process instances that have finished execution. In this work we assume that the simulation experiments are being carried out on 'as-is' process models for which historical data is available. A function has been created which extracts the historical data for a specification from the workflow engine and exports it in the *M*ining *XML* (MXML) log format. Some sample data for the credit application example is shown in Figure 3(a). This historical data is used for mining information about case arrival rates and distribution functions for the data values used in the future simulation experiments.

Similarly, the YAWL workflow system gives access to the organizational model through a function which extracts all available role and resource data in an organization and exports this information in the XML format required by ProM. Some sample data with the roles of clerk and manager are shown in Figure 3(b). This information is used to identify available roles and resources that are relevant for a given specification.

Generating the Simulation Model From the (1) extracted workflow specification, (2) the newly extracted organizational model, and (3) the event log file, we can now generate a simulation model that reflects the process as it is currently enacted. The direct usage of design information avoids mistakes that are likely to be introduced when models are constructed manually, and the automated extraction of data from event logs allows the calibration of the model based on actually observed parameters.

To generate the model, four basic steps need to be performed within ProM:

1. The workflow and organizational model and the event log need to be imported from YAWL and analyzed.



(a) A log entry for the completion of activity 'receive application' carried out by resource MoeW with loan amount $550

(b) An excerpt from an organizational model with roles and resources, where resource JonesA has role 'clerk'

Fig. 3. Part of an organizational model and historical data extracted from the workflow engine

2. Simulation-relevant information from the organizational model and log analysis needs to be integrated into the YAWL model.
3. The YAWL model must be converted into a Petri net model (because our simulation tool is based on Coloured Petri Nets).
4. Finally, the integrated and converted model can be exported as a Coloured Petri Net (CPN) model.

We can then use the CPN Tools system [10] to simulate the generated model. However, to produce useful results we do not want to start from an empty initial state. Instead we load the current state of the actual YAWL system into the CPN Tools for simulation.

Loading the Current State To carry out simulation experiments for operational decision making purposes (the 'fast forward' approach) it is essential to include the current state of the workflow system. This allows us to make use of the data values for the current cases as well as the status of the work items for current cases within the simulation experiments. A new function has been created to extract current state information for a running specification from the YAWL workflow engine and to export this information as a CPN Tools input file (see Figure 4).

The following information is obtained about the current state and is introduced as the initial state of a simulation run.

– All the running cases of a given specification and their marking.
– All the data values associated with each case.

```
fun getInitialCaseData() = [(41, {loanAmt = 1500,completeApp = false,decideApp = false}),
            (40, {loanAmt = 0,completeApp = false,decideApp = false}),
            (39, {loanAmt = 500,completeApp = false,decideApp = false})];
fun getNextCaseID() = 42;
fun getInitialTokensExePlace(pname:STRING) = case pname of
            "TASK_check_for_completeness_4`E"=>[(41,"-154","JonesA")] | _ => empty;
fun getInitialTokens(pname:STRING) = case pname of
            "Process`COND_c2_15"=>[(39,"-43200")] | "Overview`Start"=>[(40,"-155")] | _ => empty;
fun getBusyResources() = ["JonesA"];
fun getCurrentTimeStamp() = "1205203218";
fun getTimeUnit() = "Sec";
```

Fig. 4. CPN Tools input file with initial state information. Several cases are in different states in the system. For example, application No. 41 is currently being checked by JonesA for completeness, and has a run time of 154 sec, i.e., ca. 2.57 min

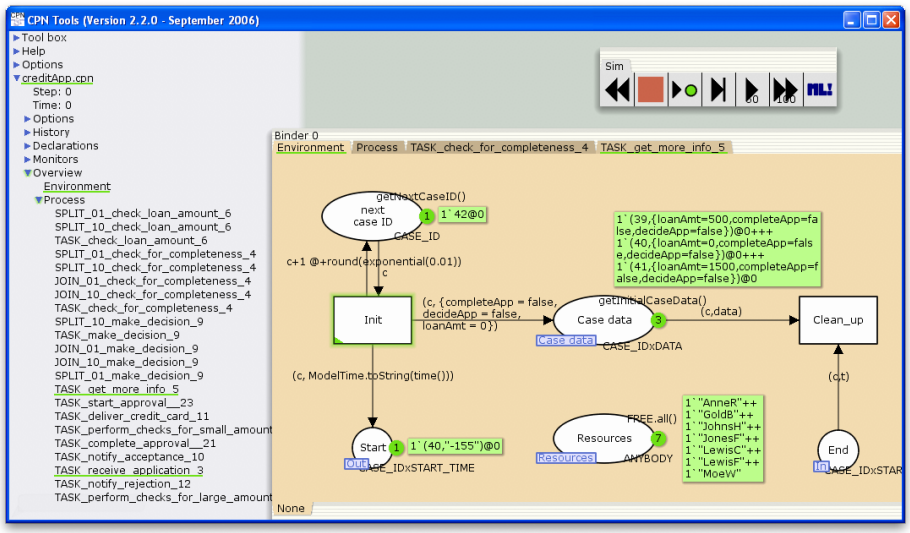


Fig. 5. The generated CPN model after loading the current state file

- Information about enabled work items.
- Information about executing work items and the resources used.
- The date and time at which the current state file is generated.

When the empty initial state file of the generated simulation model is replaced with the file depicted in Figure 4, tokens are created in the CPN model that reflect the current system status (see Figure 5). For example, among the three *Case data* tokens is the data associated with application No. 41. The resource JonesA is currently performing a check activity on this case and hence, it does not appear in the list of free resources.

We now follow the scenario described in Section 3 for simulation experiments, i.e., due to a promotion 150 cases are in the system. We load the state file containing these 150 cases into the model and perform simulation experiments

for the coming two weeks. We also add more resources to the model and observe how this influences the backlog and the throughput times for processing credit card applications within this time horizon.

Analyzing the Simulation Logs We simulate the process from the generated CPN model for four different scenarios:

1. An empty initial state. ('empty' in Figure 6)
2. After loading the current state file with the 150 applications that are currently in the system and no modifications to the model, i.e., the 'as-is' situation. ('as is' in Figure 6)
3. After loading the current state file but adding four extra resources (two having the role 'manager' and three having the role 'clerk'), i.e., a possible 'to-be' situation to help clear the backlog more quickly. ('to be A' in Figure 6)
4. After loading the current state file and adding eight extra resources (four having the role 'manager' and six having the role 'clerk'). ('to be B' in Figure 6)

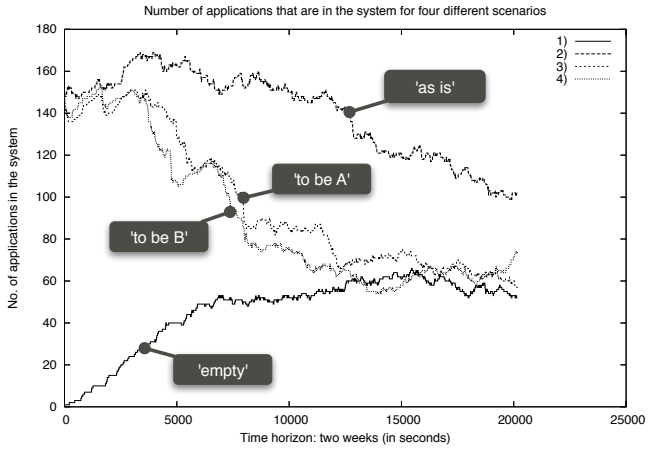


Fig. 6. Number of applications in the simulated process for the different scenarios. While the scenario with the empty state has initially 0 applications, the other scenarios are initialized by loading 150 applications from the current state file

We can see the difference among these four scenarios in Figure 6, which depicts the development of the number of cases (i.e., applications) in the workflow system over the coming two weeks for an example simulation run per scenario. In the case of Scenario 1 the simulation starts with having 0 credit card applications in the system. This does neither reflect the normal situation nor does it capture our current backlog of cases. Only after a while, does this simulation

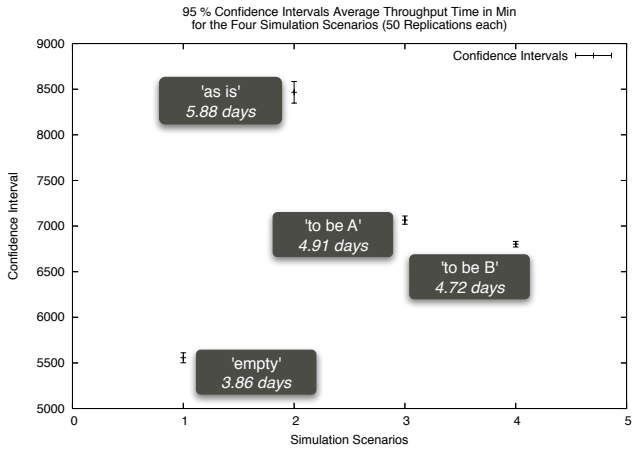


Fig. 7. Simulation run showing the 95% confidence intervals of the throughput times for the different simulation scenarios. The length of the confidence interval indicates the degree of variation

represent the normal behavior of the credit card application process (i.e., with ca. 100 applications arriving per week). The other three scenarios load a defined initial state, which contains the 150 applications that we assume to be currently in the system. Furthermore, one can observe that in the scenarios where we add extra resources to the process, the case load decreases more quickly to a normal level than without further intervention. However, the scenario 'to be B' does not seem to perform much better than the scenario 'to be A' although twice as many resources have been added. This way, we can assess the effect of possible measures to address the problem at hand, i.e., we can compare different 'what-if' scenarios in terms of their estimated real effects.

CPN Tools has powerful simulation capabilities, which we can leverage. For example, it is possible to automatically replicate simulation experiments to enable statistical analyses, such as calculating confidence intervals for specific process characteristics. For instance, Figure 7 depicts the 95% confidence intervals of the average case throughput times based on 50 replicated simulations for each of the four simulation scenarios. One can observe that the estimated throughput time for the 'empty' scenario (i.e., based on the usual situation) is ca. 4 days, while the expected throughput time for the 'as is' scenario (i.e., actually expected based on the current backlog situation) is almost 6 days.

While CPN Tools already provides powerful logging facilities and even generates gnuplot scripts that can be used to plot certain properties of the simulated process, we also generate MXML event log fragments per case during simulation, similar to the one shown in Figure 3(a) for the workflow log. These fragments can then be combined using the CPN Tools filter of the ProM*import* frame-

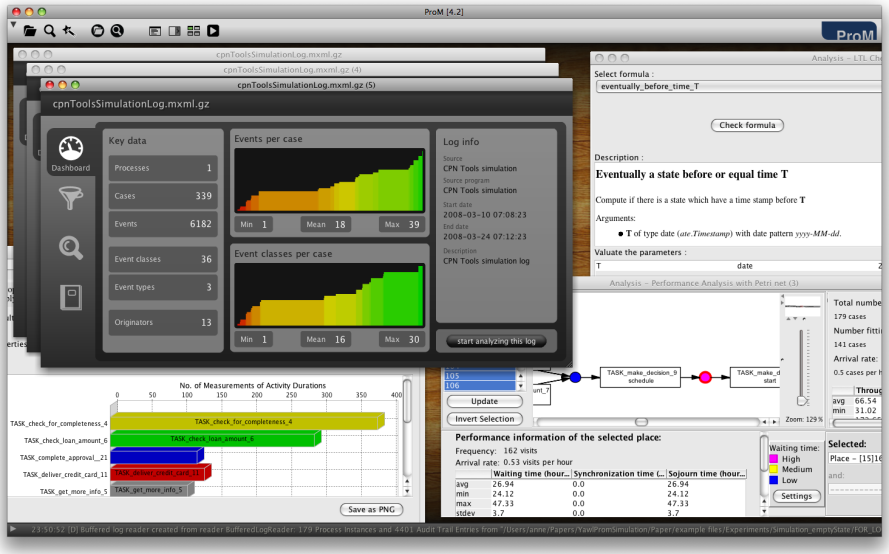


Fig. 8. The generated simulation logs can be analyzed with the same tool set as the initial workflow logs

work, which facilitates the conversion of event logs from various systems into the MXML format that is read by ProM.

The ability to use the same toolset for analyzing the simulation logs and analyzing the actual workflow logs constitutes a big advantage because the simulation analysis results can be more easily related to the initial properties of the process. In particular, since we support the loading of current cases into the initial state at the beginning of the simulation, *we can easily combine the real process execution log ('up to now') and the simulation log (which simulates the future 'from now on') and look at the process in a unified manner* (with the possibility of tracking both the history and the future of particular cases that are in the system at this point in time).

Figure 8 shows a screenshot of ProM while analyzing the simulation logs generated by CPN Tools. Various plug-ins can be used to gain more insight into the simulated process. For example, in Figure 8 the Log Dashboard (top left), the Basic Statistics plug-in (bottom left), the Performance Analysis plug-in (bottom right), and the LTL Checker (top right) are shown. The former two provide a more general overview about the cases and activities in the process, whereas the Performance Analysis plug-in finds bottlenecks (e.g., in Figure 8 a bottleneck for starting the activity 'Make decision' is highlighted), and the LTL Checker can be used to verify specific properties of interest (e.g., "How many cases could be processed until they are in the stage where a decision can be made in under 3 days?").

4 Tutorial

In this tutorial section, we will demonstrate the creation of a simulation model for a YAWL process based on the running example. As described earlier, the resulting simulation model consists of two parts: (1) a CPN model that represents the process (based on specified and discovered characteristics), which is more or less static, and (2) a separate SML file containing information about the state of currently running cases and available resources, which can be exported from the YAWL engine and will be read by CPN Tools to create tokens that represent this non-empty initial state of the simulation experiment. This separation allows the most up-to-date current state file to be used without the need to change the simulation model.

Figure 9 depicts the landscape of tools and information sources that are involved in the creation of the simulation model for the YAWL process. In the top left corner one can see the process designer who initially creates a YAWL process specification that is to be enacted on a *YAWL engine*. Note that in this tutorial we do not provide any details about the creation and deployment of processes as this is covered by the documentation of the YAWL workflow system[3]. Instead, we demonstrate how to create a simulation model for a running process, and allow you to try this yourself using the files from the example[4]. In Figure 9, one can see that the *YAWL editor* has access to a *Resource database*, which contains information about the relations between roles and resources in the organization. This is needed to support resource allocation mechanisms in the YAWL editor, to associate tasks with roles (i.e., specify which people are allowed to perform which tasks in the process). This organizational model is also used by the YAWL engine during enactment when newly created work items are to be offered to users of the workflow system. Finally, the YAWL engine records *log data* during the execution of each case (i.e., process instance).

All these data sources (the process specification, the organizational model, and the execution log containing historic data) are relevant for the creation of a simulation model that reflects reality as closely as possible. The ProM framework[5] allows for the creation of a CPN model based on different data sources. Furthermore, knowledge about the current state (operational characteristics) of the process, which is available from the YAWL engine itself, is needed for short-term simulation tasks. By making this current state information available through an SML file, it can be loaded dynamically into the CPN model to let its marking reflects the current state of the process. Finally, one possibility to obtain information about the simulation itself is to generate simulation logs that in turn can be analyzed using the ProM framework.

[3] User manuals and technical documentation for YAWL are available from the public website at *http://sourceforge.net/projects/yawl/*.

[4] You can download the example files to follow the steps shown in this document from the Tutorial page at *http://www.processmining.org/*.

[5] The ProM framework including source code and documentation can be freely downloaded from *http://prom.sf.net/*.

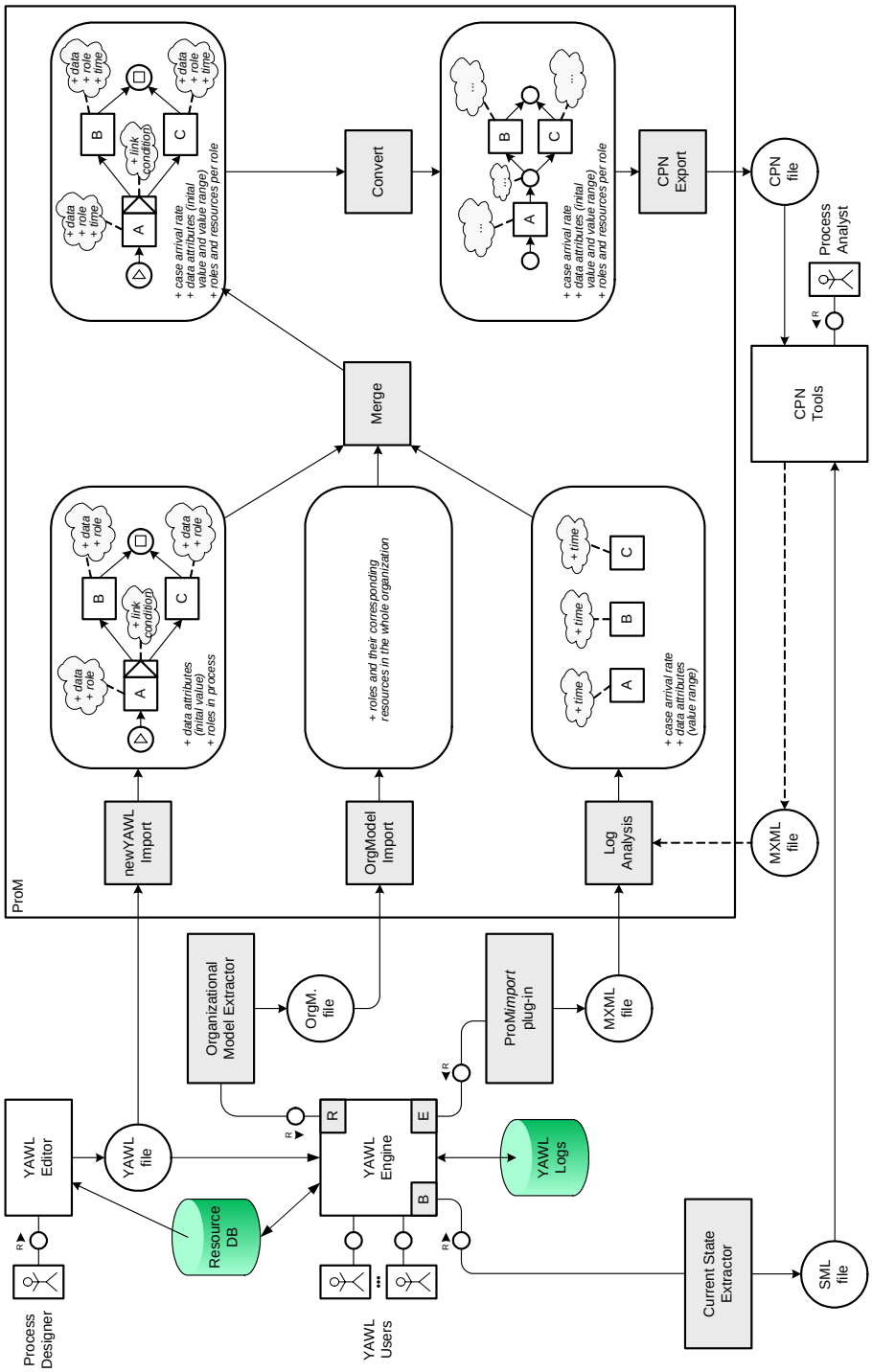


Fig. 9. Overview of the involved tools and information sources

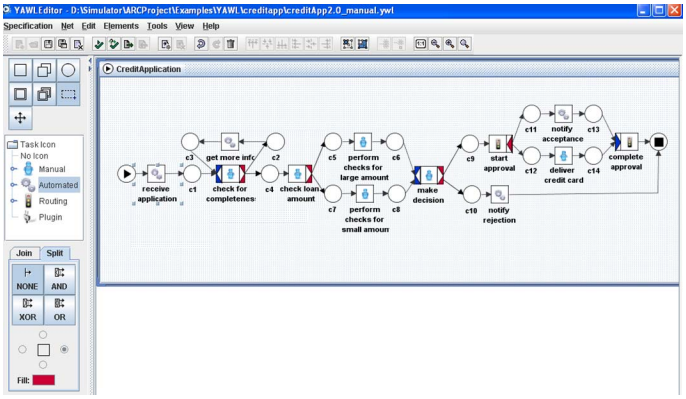
To extract all the simulation-relevant information from YAWL in a form that can be interpreted by ProM and CPN Tools, a number of steps need to be taken. They are described in more detail in Section 4.1. Then, the creation of the actual CPN model after importing and merging the different data sources is explained in Section 4.2. Afterward, the incorporation of current-state information in the CPN model is discussed in Section 4.3, and the generation of MXML logs from CPN Tools is described in Section 4.4. Finally, the features and limitations of the current implementation are summarized in Section 4.5.

4.1 Extracting Simulation Information from YAWL

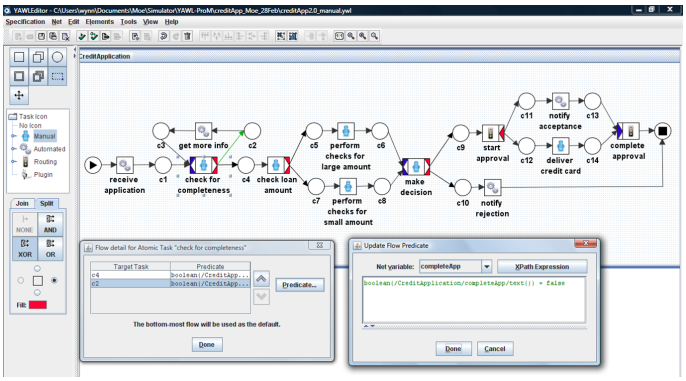
In this section, we first describe briefly how to construct a YAWL workflow model in the YAWL editor. It is followed by the discussion on how to retrieve simulation relevant information from the YAWL workflow system for simulation experiments.

Creating and Deploying the YAWL Engine File Here, we briefly mention the notation used in YAWL. In YAWL, tasks are depicted as boxes, conditions (or places in Petri net terminology) as circles, and the control flow is modelled using directed arrows between tasks and conditions. Tasks can be argumented with a join behavior on the left of the task symbol and a split behavior on the right side of the task symbol, which is one of XOR, AND, and OR. If the split and join behaviors of a task are not explicitly specified, it should be considered as having a default XOR-join behavior and an AND-split behavior. For instance, the check loan amount task is modelled with an XOR-split behavior and a (default) XOR-join behavior. Similarly, the start approval task has an AND-split behavior and a (default) XOR-join behavior. The starting point in the process is depicted using an input condition (which is represented by a circle with a filled triangle inside) and the end point is depicted using an output condition (which is represented by a circle with a filled square).

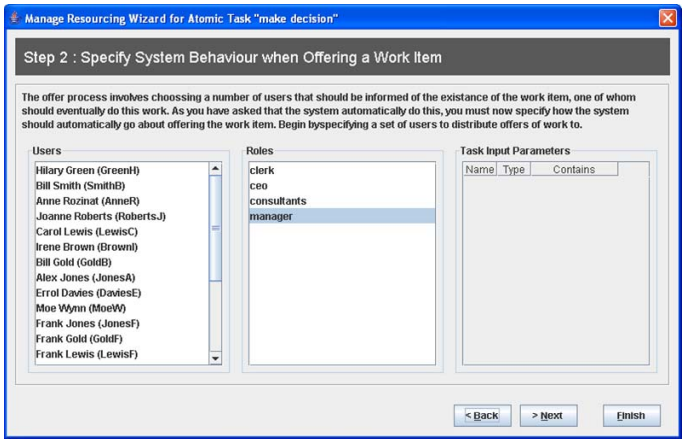
In addition of specifying the control flow requirements of a process model, it is also possible to specify the data and the resource requirements of the process using the YAWL editor as shown as Figure 10. The data perspective in YAWL is modelled using XML technologies. For instance, data attributes are modelled as XML data types, string, boolean etc. The decision points such as XOR-splits are modelled using Xpath expressions (see Figure 10(b)). In the credit application example, three variables were used to model decision points, one numeric attribute for the loan amount (loanAmt), and two boolean attributes (completeApp and decideApp). A rich resource behavior of a YAWL task can also be specified using the information available from an organizational database (see Figure 10(c)). For simulation purposes, we chose to assign resources based on the role-based allocation mechanism to align the model with the simulation capabilities of the CPN tools. Clerk and Manager roles are used in the example process to demonstrate the role-based allocation strategy. For instance, we specify that only a resource with a manager role can work on the Perform checks for large



(a) Specifying the control flow behavior



(b) Specifying the decision based on the loan amount in xpath



(c) Specifying the resource requirements of the make decision task (manager role)

Fig. 10. Creating a workflow specification in the YAWL editor

amount and Make decision tasks. After the process designer has specified all the requirements of the process in the editor, the specification is exported from the editor as an XML file. This file is used by the YAWL execution environment (YAWL engine) to enact the workflow.

Converting the YAWL Logs into MXML Data In the YAWL workflow system, event logs are being created whenever an activity is enabled, started, completed or cancelled together with the time when it is enacted and who has enacted this event. The logs are also kept for the data values that have been entered and used throughout the system. Therefore, it is possible to easily retrieve historical data about a specification that has been executed before. A function has been created which extracts the historical data for a given specification from the engine (through Interface E) and exports this information in the MXML mining log format. This conversion function is made available through the ProM*Import* framework and a screenshot of the user interface is shown in Figure 11.

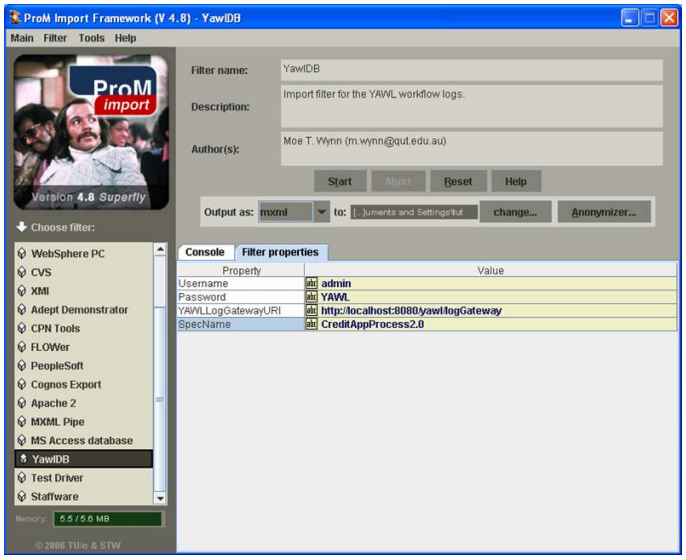


Fig. 11. A screenshot of the YAWL log filter

Exporting the Organizational Model from the Resource DB Similarly, the YAWL workflow system exposes the organizational model data through Interface R. A function has been created to extract the organizational model data which contains all available roles and resources in the resource DB and to export this information in an XML format required by ProM (see Figure 12).

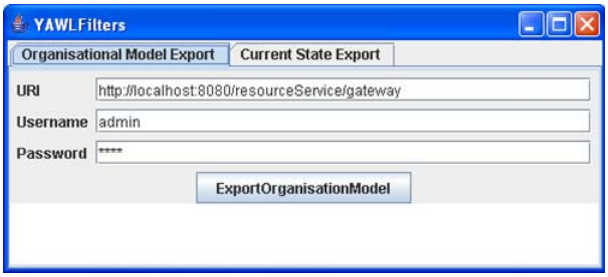


Fig. 12. A screenshot of the YAWL Organisation Model filter

Exporting the Current State from the YAWL Engine A new function has been created to extract the current state information for a running specification in the YAWL workflow system using Interface B and to export this information as a CPN tool input file (Figures 13 and 4). All three functions require an active connection to a running YAWL engine (local or remote) to request the required data.

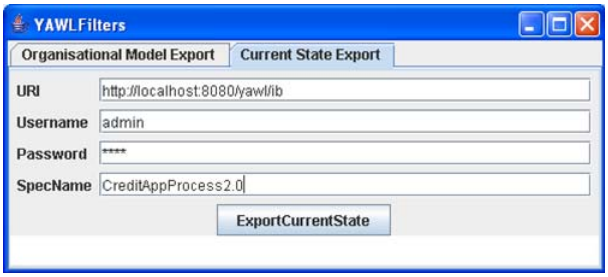


Fig. 13. A screenshot of the YAWL Current state filter

4.2 Creating a Simulation Model in ProM

As one can see in Figure 9, we have three data sources from which we build the simulation model: the YAWL engine file, the MXML execution log, and the organizational model file. These sources need to be imported, and merged into a YAWL-based high-level model. Note that with high-level information we refer to process information beyond the pure control-flow, i.e., additional information like data and time that can be either attached to the process as a whole, or to certain elements in the process (see for example the items in the clouds in the overview picture in Figure 9, which reflect high-level information attached to activities and choices). The merged model is then to be converted into a Petri-net based high-level process, which can be exported to CPN Tools.

In the remainder of this sub section we describe in detail how to import, combine, and transform the simulation-relevant information, so that a CPN model reflecting the running YAWL process can be generated from it.

Importing the YAWL Engine File To open the YAWL engine file (in the following also called YAWL specification), press the left-most button in the ProM toolbar, locate the YAWL specification on your file system, and then choose *newYAWL* from the list of import types (see Figure 14).

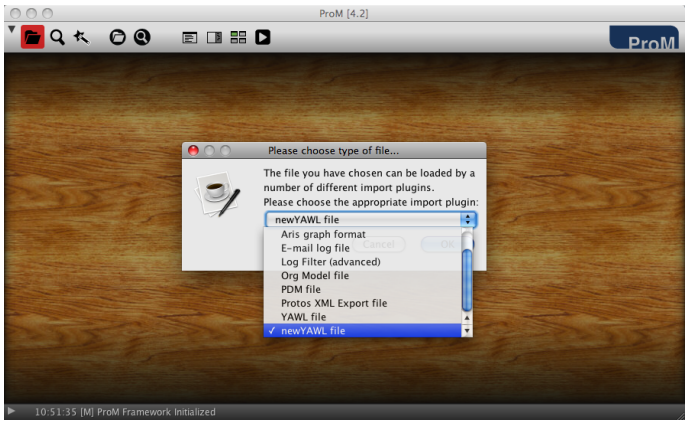


Fig. 14. Opening the YAWL engine file in ProM

As can be seen in Figure 9, the information we can get from the YAWL engine file covers a YAWL process model including roles at tasks, data flow, and link conditions at choice points in the process. When importing the YAWL engine file into ProM, we create a YAWL-based high-level process, which is immediately visualized using the `View/Edit High Level Process` plug-in in ProM. This plug-in provides an editable view on the non-control flow characteristics of the YAWL specification.

The different views are discussed in the following sub sections. Note that currently only non-hierarchical YAWL models are supported to create simulation information. That is, if an imported YAWL process contains sub processes, then only the top-most level of the hierarchy is evaluated and a sub process may be seen as only one step in the model. For a detailed list of limitations and assumptions please refer to Section 4.5.

Data Variables Figure 15(a) and Figure 15(b) show the data view of the high-level process. We can see the names of the three data attributes in the example process in the table to the left, which are *loanAmt*, *completeApp*, and *decideApp*. While both attributes *completeApp* and *decideApp* are of type boolean, i.e., their possible values are only “true” and “false”, the attribute

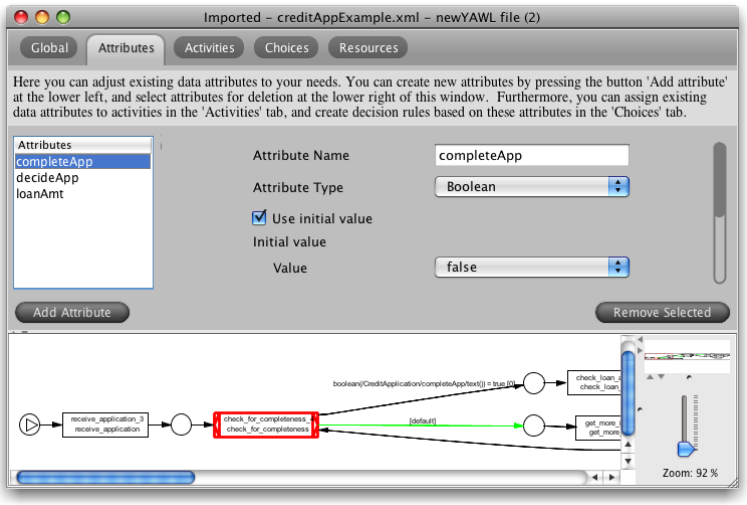
loanAmt is of type numeric. Note that double-clicking an attribute in the table to the left will highlight all activities in the process that provide this data item. For example, in Figure 15(a) the attribute *completeApp* was clicked, and as a consequence the task "Check for completeness", which outputs this data item, is marked in the YAWL model visualization.

Next to the name and the type of an attribute we also get the initial value from the YAWL specification. For example, for the *loanAmt* attribute displayed in Figure 15(b) we see the initial value 0. If an explicit initial or default value is available, then this value will be used when the case data is initialized at the beginning of the simulation. If no initial value is provided by the YAWL specification (which is, for example, the case for variables of the type *inputParam*), a random value can be generated upon initialization.

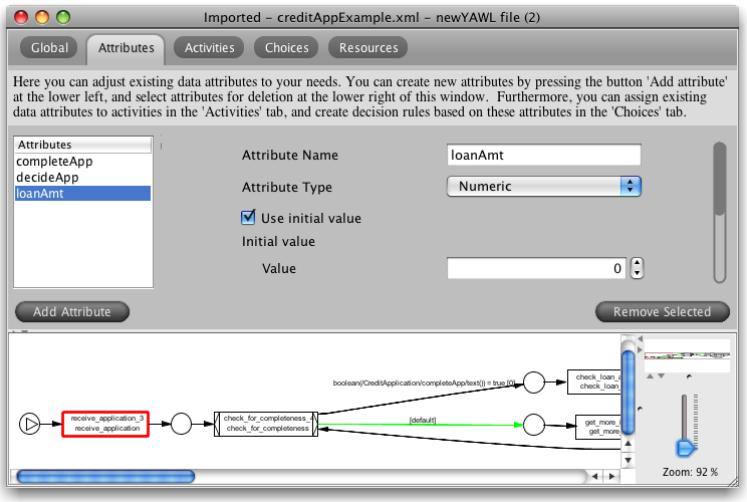
However, for a realistic simulation it is also important to know what are typical actual values (value range distribution) for an attribute. For example, if we simulate our example process for many cases with a low amount (in which case it suffices to perform a quicker check task), but in reality we deal with almost only high amount claims (in which case we need to perform a lengthy check procedure), then the simulation outcome is likely to be overly optimistic. Typical actual values cannot be obtained from the YAWL specification itself, but we will get this later from the log analysis (see Section 4.2).

Activities and Role Assignments Figure 16(a) shows the activity view of the high-level process. We can see the names of the activities in the example process in the table to the left. Double-clicking any of them will show the corresponding activity information, and also highlight the belonging YAWL task in the visualization below. From the YAWL engine file we get information about the activities with respect to their relation to data attributes and roles. For example, in Figure 16(a) one can see that the activity "Check for completeness" outputs the data attribute *completeApp,* and that the *Role ID: 2* is required to perform this task. However, we do not obtain information about, for example, execution time and waiting time statistics for the activities. For this, we need to analyze the execution log, which is described in Section 4.2.

Figure 16(b) shows the resource view of the high-level process. From the YAWL specification we can obtain the roles that are relevant in the context of this process. For example, in Figure 16(b) we can see that our example process requires two different roles, which are *Role ID: 1* and *Role ID: 2*. These are IDs referring to actual roles in the Resource DB depicted in Figure9. However, we do not obtain any information about the organizational structure itself, such as which resources actually belong to those roles. Observe that the set of resources belonging to these roles in Figure 16(b) is empty. Also, we do not know the names of these roles. If we would look up the IDs in the Resource DB, we would find out that the role with ID *1* corresponds to the *manager* and the role with the ID *2* corresponds to the *clerk* role.

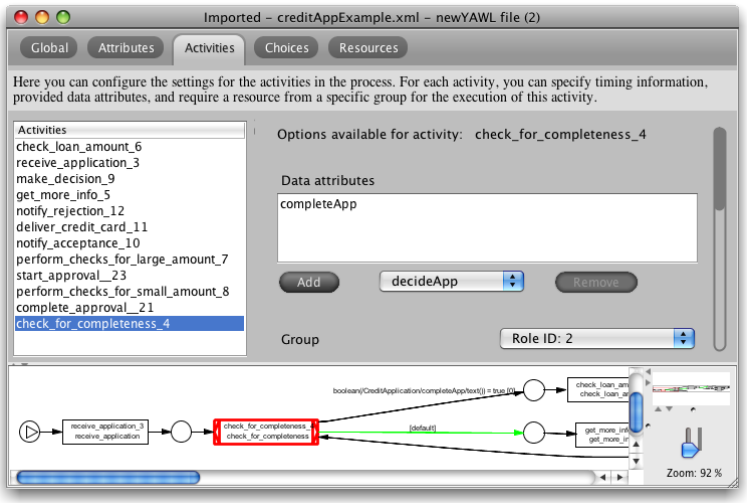


(a) Boolean data attribute view

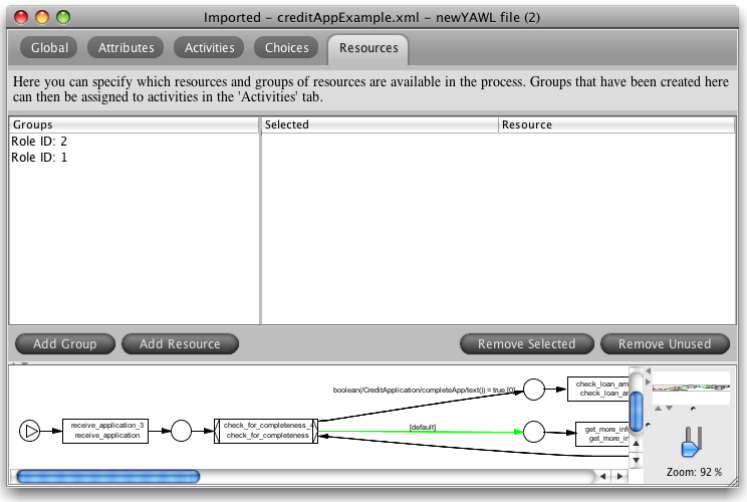


(b) Numeric data attribute view

Fig. 15. Data variables and their initialization values are imported together with the YAWL specification



(a) Activity view



(b) Resources view

Fig. 16. Output data variables and role assignments per activity are imported together with the YAWL specification

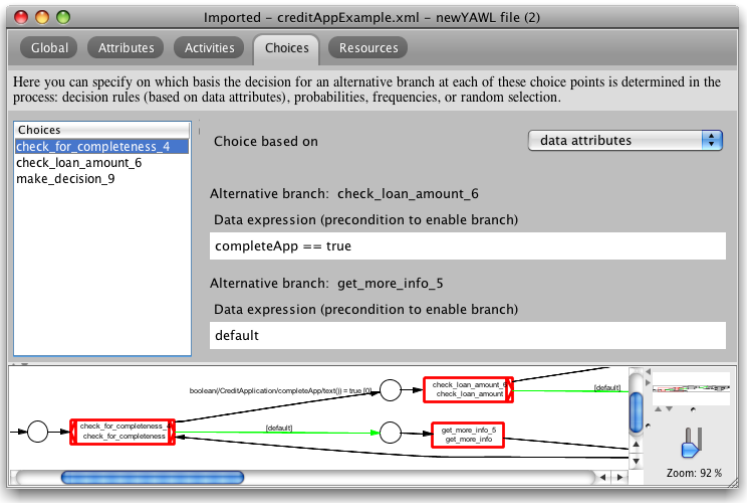
Because the number of available resources per role significantly influences the concurrent processing power of a process, we need this organizational information for our simulation model. Section 4.2 describes how organizational data that was obtained from the YAWL workflow system can be loaded into ProM.

Link Conditions Figure 17 shows the choice view of the high-level process. We can see the names of the choice points in the example process in the table to the left, which are "Check for completeness", "Make decision", and "Check loan amount". They all correspond to all XOR-split tasks, but also conditions with more than one outgoing arc would constitute a choice point in a YAWL process. If you double-click one of these choices in the table, the corresponding choice node in the YAWL process is highlighted together with the target activities (i.e., the activities which may be enabled by this choice)[6]. We can obtain the enabling conditions for each of these target activities from the flow predicates in the YAWL specification. In Figure 17(a) the choice point "Check for completeness" and the enabling conditions for the two targets "Check loan amount" and "Get more info" are displayed. Figure 17(b) depicts a screenshot of the information about the choice point "Check loan amount" with the two target activities "Perform checks for large amount" and "Perform checks for small amount".

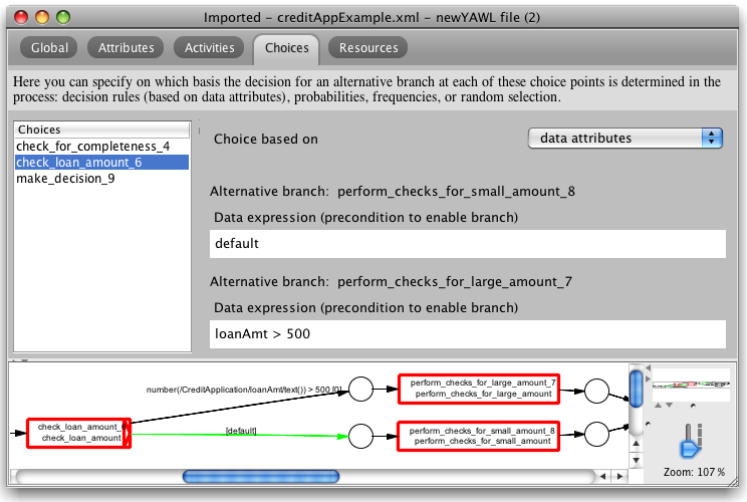
Note that while YAWL here allows for arbitrary XPath expressions, we currently only read simple boolean conditions over data attributes in the process. However, we do evaluate the default branch semantics and respect the order rank provided for each flow predicate. The order rank is used in YAWL to ensure that only one branch is enabled for an XOR split: if more than one of the descending flow conditions evaluates to "true", then only the branch with the lowest order rank will be chosen. The order rank information is stored with the expressions, and is used later on to generate consistent preconditions in the CPN Export. Refer to Section 4.5 for a detailed overview of features and limitations in the current implementation.

Analyzing the YAWL Execution Log In the next step, we want to analyze the YAWL execution log. To open the MXML log file, press the left-most button in the ProM toolbar, choose the file from the right location, and leave the import type as *MXML Log reader* (see Figure 18). The resulting log window in ProM

[6] Be aware that the decision point concept employed in ProM is slightly different from the one in YAWL. In ProM, where - coming from a mining perspective - generally only task occurrences can be observed (but no explicit state information is available), it is natural to define data dependecies in terms of pre-conditions for activities. YAWL comes from a specification viewpoint, where it is possible to define complex routing patterns that enable or not enable particular branches (regardless of possible further conditional branching). So, in YAWL enabling expressions are attached to arcs rather than to activities (and tokens will be produced for the output condition nodes depending on whether the flow predicate is fulfilled or not). Therefore, when we import the YAWL specification, we automatically map the initial branching conditions onto pre-conditions for the corresponding (target) activities.



(a) Link condition "Check for completeness"



(b) Link condition "Check loan amount"

Fig. 17. Link conditions at XOR split tasks are imported together with the YAWL specification

is shown in Figure 19(a). Within this log dialog, the dashboard and the log summary already provide an overview and simple frequency statistics about cases and activities in the logged process. Furthermore, the log can be inspected and filtered.

Generally, the event log is a valuable source of information and enables various kinds of analysis. For example, we can use it to discover a process model, or check the conformance of the real process to some a-priori model. For our simulation model we specifically seek information about the case arrival rate, value range distributions for data attributes, and observed execution times at tasks in the process (cf. Figure 9). The extraction of these characteristics from an execution log provides us with the opportunity to further approximate the real process.

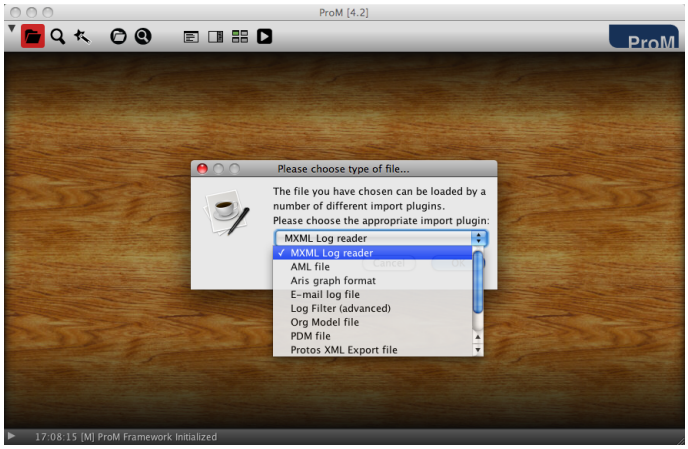
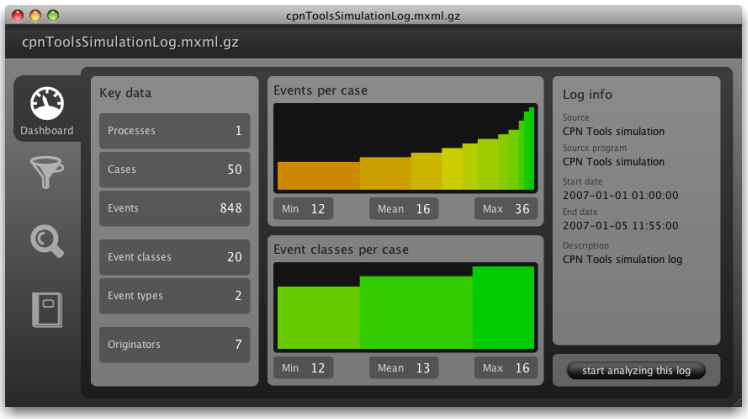


Fig. 18. Opening the MXML log file in ProM

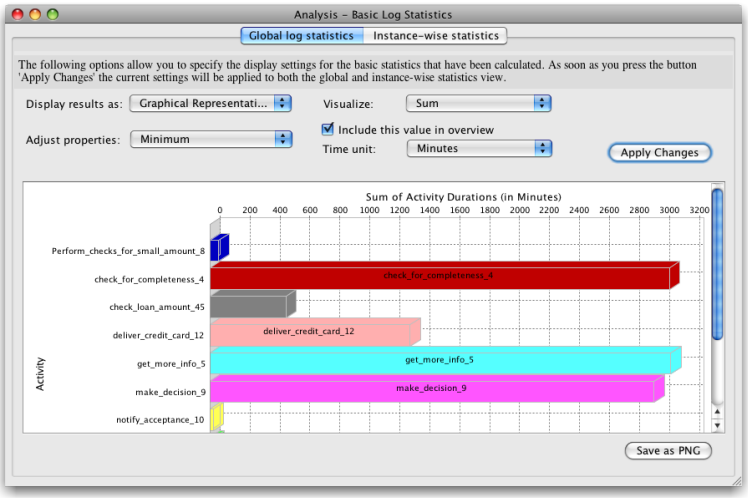
To obtain these characteristics, we can analyze the log with the `Basic Log Statistics` plug-in in ProM. It can be invoked via choosing *Analysis* → *Basic Log Statistics* from the menu while the window shown in Figure 19(a) is in the foreground. Alternatively, you can press the Action trigger ("Play") button in the toolbar, or press <Command/Control> + <D>, and start typing until the plug-in "Basic log statistics" is selected. Then, hit the $< RETURN >$ key.

The Basic Log Statistics plug-in allows to view simple time and data attribute statistics about the process both in a table and a graphical view as depicted in Figure 19(b). In addition, it creates an activity-based high-level process, which can be viewed with the plug-in `View/Edit High Level Process`. The relevant views are discussed in the following sub sections.

Case arrival rate Figure 20(a) shows the global characteristics of the example process, among which is the discovered case generation scheme. Information about the average number or arriving cases is crucial for simulation as it



(a) Dashboard of the Log dialog



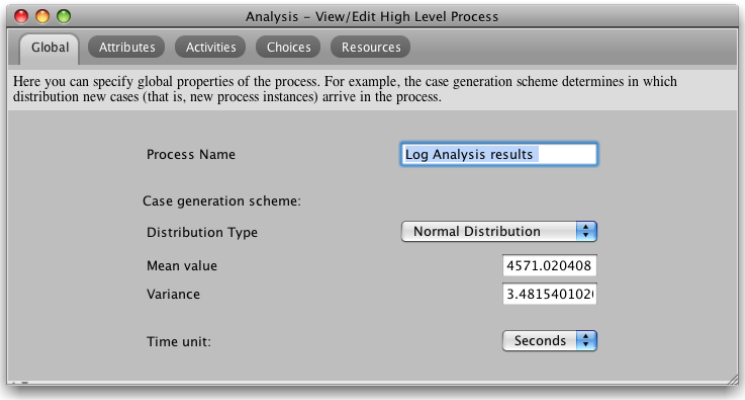
(b) Analyzing execution times with the Basic Log Statistics plug-in

Fig. 19. Analyzing the execution log in ProM

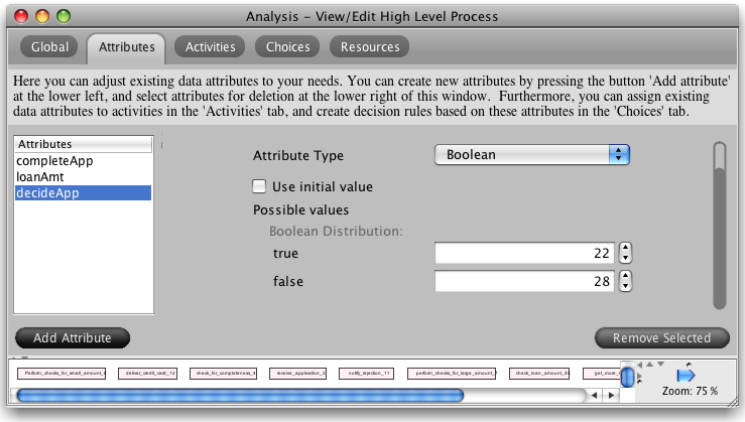
influences the workload in the system. Basing this parameter on real process execution data makes the simulation model more realistic.

Note that the number of arriving new cases is also a simulation parameter that one might want to modify for short-term simulation. Imagine, for example, a company that expects twice as many applications as usual before the Christmas period.

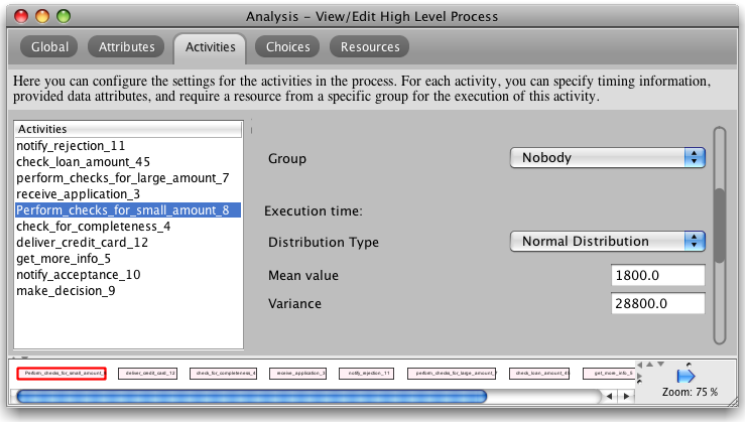
Data attributes and their value range Figure 20(b) depicts the discovered value range distributions for the data variables of the example process. This is supported for boolean, numeric and nominal attributes. Basing the simulation on typical value distributions obtained from log analysis also helps



(a) Case arrival rate



(b) Data value range statistics



(c) Execution times of activities

Fig. 20. Viewing the extracted process characteristics

to further approximate the real-world process. For example, if many applications are typically not complete, then more iterations in the process are needed on average until it can be completed.

Activities and their execution times Figure 20(c) shows the activity view for the high level process extracted from the execution log. We can see that the execution times of the activities (here for "Perform checks for small amount") are now characterized by a distribution that was discovered from the log.

Importing the Organizational Model Finally, we want to import the organization model file that has been extracted from the YAWL workflow system. To open the OrgModel file, press the left-most button in the ProM toolbar, locate the file, and choose *Org Model* from the list of import types (see Figure 21).

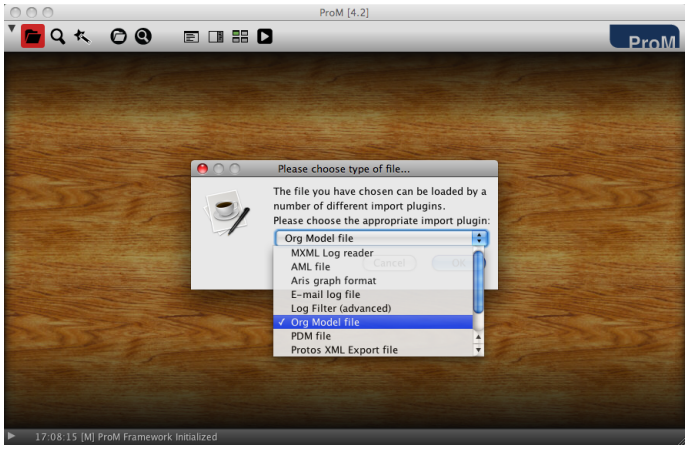
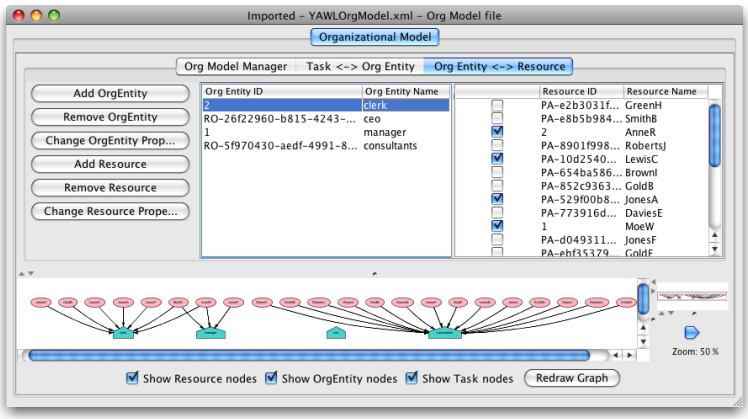


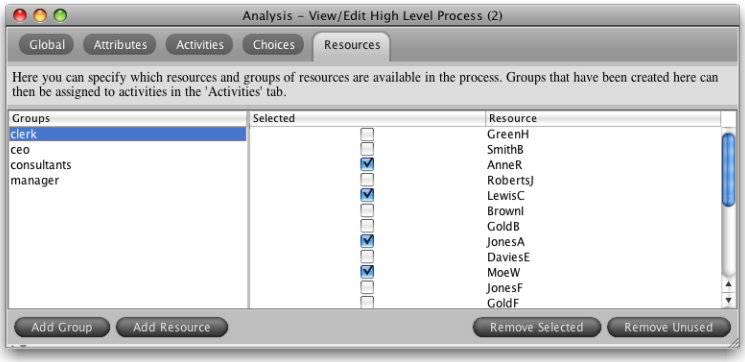
Fig. 21. Opening the organizational model file in ProM

Then, leave the option *Open without log file* selected, and press "OK". The resulting view is shown in Figure 22(a). When importing the OrgModel file into ProM, we create a high-level process that only contains this organizational information, and which again can be inspected via the `View/Edit High Level Process` plug-in. The resulting resources view in the high-level process is shown in Figure 22(b).

As can be seen in Figure 9, the information we can get from the OrgModel file covers the relationship between all roles and resources in the whole organization. For example, in our credit card application process the roles *ceo* and *consultant* are not needed. During the merging phase of the various inputs, we will discard those roles that are not relevant for the process to be simulated.



(a) Resulting view after importing the organizational model



(b) The imported resource information in the high-level process

Fig. 22. The available roles and resources and their relationships are imported with the organizational model

Merging the Different Sources Now that we have extracted simulation-relevant information from the YAWL specification, the organizational model, and the execution log, we need to merge all that together (cf. Figure 9). To do this, there is a plug-in called `Merge Simulation Models` in the ProM framework.

It can be invoked by pressing the second left-most button in the ProM toolbar, and then selecting "Merge Simulation Models" from the list of plug-ins to the left. On the right hand side the input models to be merged can now be selected. Add one more input, and choose the imported YAWL model, the imported organizational model, and the log analysis results, respectively. Then, press the "Next >>" button and choose the YAWL model as the reference model like shown in Figure 23. This means that the simulation-relevant information from the organizational model and the log analysis will be integrated in the YAWL-based simulation model. Afterward, two dialogs will appear where the high-level

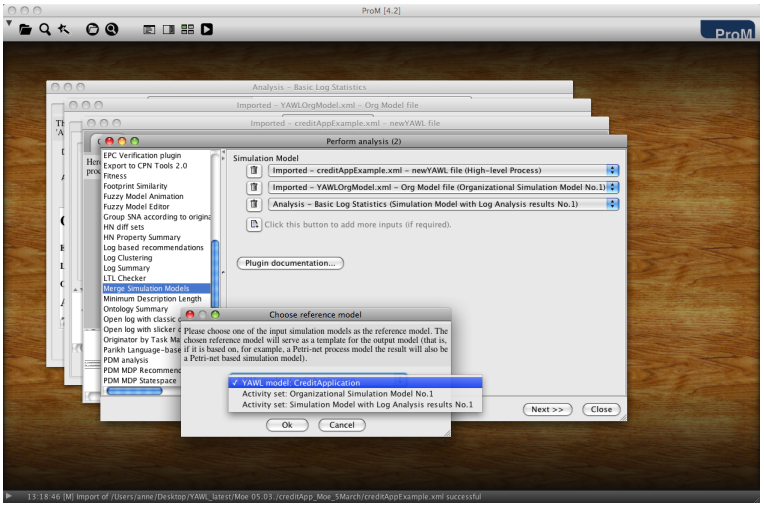


Fig. 23. Invoking the Merge Simulation Models plug-in in ProM

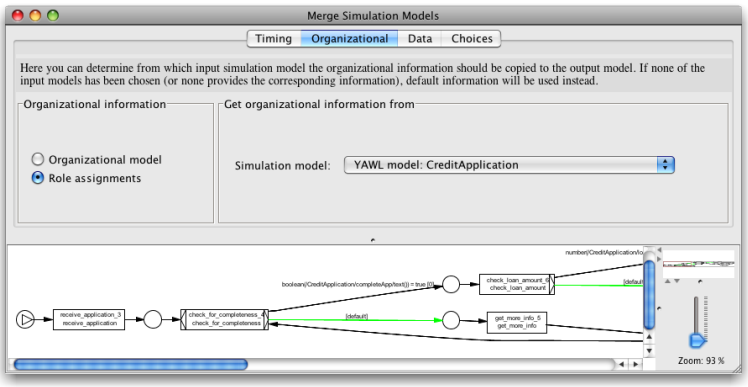
activities of the input models to be integrated will be mapped onto the reference model. You can simply accept the default mapping, which is established based on the names of the activities.

Now, the plug-in creates a new high-level model based on the YAWL template model, and the resulting view is shown in Figure 24(a). It automatically makes a pre-selection of which pieces of simulation-relevant information are to be taken from which model. We do not need to change anything. For example, in Figure 24(a) one can see that the *role assignments* are obtained from the YAWL specification. If you select the *organizational model* item, you will see that this part of the model is taken from the imported organizational model.

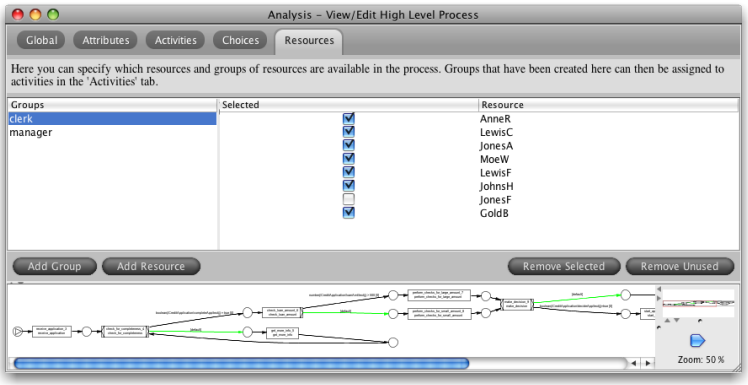
To inspect the details of the merged high-level process, we can again use the `View/Edit High Level Process` plug-in. You will see that the merged model now contains all previously extracted elements: case arrival rate, data variables including initial values and value range distributions, activies including their output variables, role assignments and execution time distributions, and link conditions, and the roles and resources from the organizational model.

One thing we can do now because we incorporated both the role assignments for the activities and the organizational model is to dispose of all the groups and resources that are not relevant for our example process: Simply press the button "Remove unused" in the resource view. Figure 24(b) depicts a screenshot of the resource view after this has been done.

Converting from YAWL to Petri Net As can be seen in Figure 9, there is one last step that needs to be performed before we can generate the CPN



(a) Merge Simulation Models plug-in (organizational perspective)



(b) Merged process after removing all unused roles and resources

Fig. 24. The simulation-relevant information is automatically merged based on the covered perspectives

model for our example process[7]. Because the CPN Export expects a Petri net-based model as input, the YAWL process needs to be translated into a Petri net process. However, of course we want to keep all the high-level information that was extracted and merged together linked to the activities and choices in the process. This can be achieved by invoking a conversion plug-in called `HLYAWL to HLPetriNet`. Again, you can either use the menu or the Action trigger while the merged model from Figure 24(b) is in the foreground.

The converted model view is immediately visualized using the `View/Edit High Level Process` plug-in, which is depicted in Figure 25. One can see that the link conditions that were obtained from the YAWL specification are pre-

[7] Alternatively, one could also convert the YAWL model directly after the import, merge the organizational model and the log analysis results into the Petri net-based model, and then invoke the CPN Export from there.

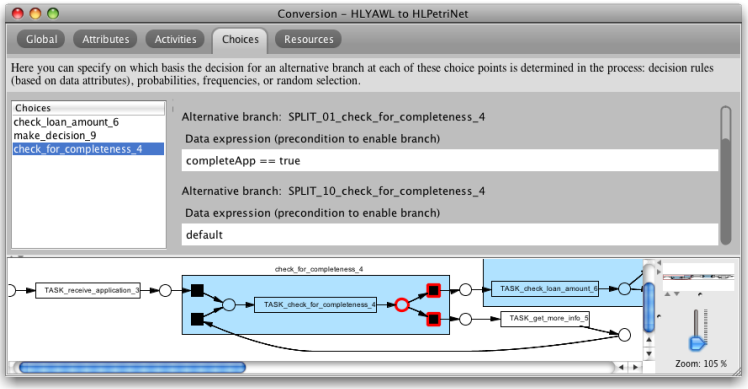


Fig. 25. Viewing the converted model

served, and are now associated to the invisible (i.e., routing) transitions that represent the XOR split after the conversion[8].

Exporting the CPN Model Now we have everything that is needed to generate a CPN model for our example process: Based on the Petri net-based high-level process shown in Figure 25, the `Export to CPN Tools` 2.0 plug-in can be invoked. Again, simply press the action trigger and start typing "Export to CPN Tools 2.0". If you want to use the menu, be aware that—for some technical reason—the plug-in is not an export but an analysis plug-in. Therefore, it will not appear in the export but the analysis menu.

Figure 26 depicts the configuration screen of the CPN Export plug-in. On the left side, one can choose which perspectives should be included in the model. On the right side, one can determine which extra functionality, such as the current state support and the logging monitors, should be generated. Select all of them like shown in Figure 26. If you then press the "Export" button, and choose an export location, then the .cpn file plus a file with the extension .sml (and the same name as the .cpn file) will be generated. This .sml file represents the current state, and per default the initial state will be empty. However, you can load some real initial state reflecting the current situation in the workflow system, which is described in the next section.

[8] Note that we deliberately preserve the invisible transitions resulting from XOR splits and XOR joins, even though they might seem redundant. This is needed to preserve the moment of choice in connection with the current state: imagine that for some case the decision for which alternative branch to take was already made, but the activity has not been started yet. Reflecting this in the model would not be possible if we would remove the invisible XOR split transitions.

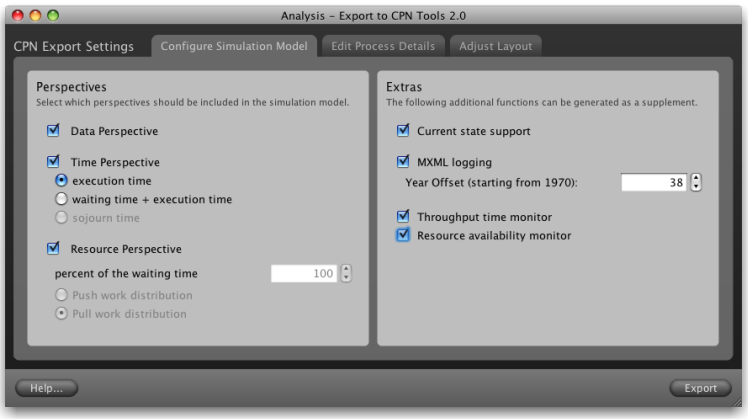


Fig. 26. Configuring the simulation model before starting the export

4.3 Loading the Current State Information into the CPN Model

To load an actual current state that was extracted from the YAWL engine before (cf. Figure 9) into the CPN model, you simply need to rename the .sml file that was extracted from the engine in such a way that it has the same name as the empty initial state file that was generated together with the CPN, and to place it into the same directory as the CPN model (i.e., replace the empty current state by the actual current state). If you then re-open[9] the generated CPN model, you will see tokens in the model that represent currently running cases.

Figure 27 shows another screenshot of the CPN model with the loaded current state file (see also Figure 5). In Figure 27 you can see the sub page of the activity "Check for completeness", where the case with the ID 41 is currently in the state of being executed by the resource "JonesA". As a consequence, the resource "JonesA" is not among the currently available resources.

4.4 Generating MXML Logs during Simulation in CPN Tools

The generated CPN already contains logging monitors that automatically create event logs during the simulation of the model in CPN Tools. The screenshot in Figure 5 depicts the simulation binder including *play*, *stop*, and *fast forward* buttons, which can be used to invoke a simulation of the process for a determined number of steps. During simulation event logs are created in a folder called *logs*, which is located in the same directory as the CPN file itself. This folder contains one file with the extension *.cpnxml* per started case. To obtain a coherent MXML log, there exists a ProM*import* plug-in called "CPN Tools", which simply bundles all the cases from the simulation in one process log.

[9] Make sure to close all open CPN Tools windows including any console windows before you open the CPN model with the newly loaded current state.

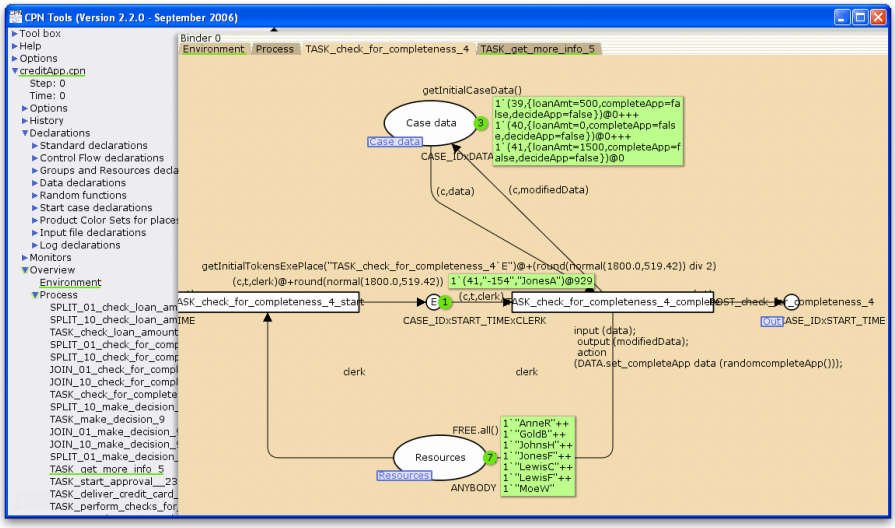


Fig. 27. Generated CPN model with loaded current state

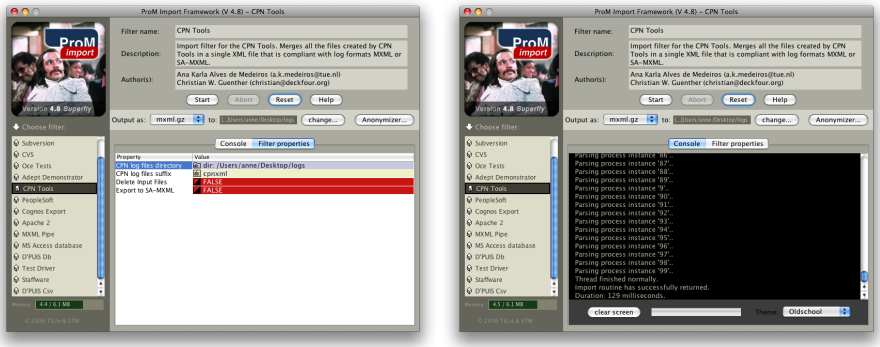


Fig. 28. ProMimport plug-in for creating MXML logs from CPN Tools simulation logs

Figure 28 depicts screenshots of the filter properties and the console output view of the "CPN Tools" plug-in in ProM*import*[10] [7]. One only needs to select the *logs* folder containing the *.cpnxml* files from the desired simulation run, and press the "Start" button. As a result, the combined MXML log will be created in the chosen output location. More information on how to use CPN Tools to create synthetic logs is provided at the ProM CPN Library Tutorial (www.processmining.org, "Tools" section).

[10] The ProM*import* framework including source code and documentation can be freely downloaded from *http://promimport.sf.net/*.

The simulation log in MXML format can now be loaded into the ProM framework to analyze the simulated process. Using the same tool set for analyzing the original execution logs as well as the simulation logs makes it easy to compare the (effects of changes in the) simulated process with the observed real-life process.

4.5 Overview Features and Limitations

This last sub section is intended to give a compact overview about features and limitations of the current implementation.

Features Building on existing tool sets such as YAWL, CPN Tools, ProM and ProM*import*, we can provide an implementation that supports most aspects that need to be considered for Business Process Simulation. The following summarizes the previously described functionality:

- YAWL models (enactable engine files generated with the YAWL editor) can be imported into ProM, whereas the defined data variables, link conditions, and role assignments for the contained task nodes are extracted and can be used for simulation.
- Organizational models can be exported from the YAWL resource database (used both by the YAWL editor as well as the engine enacting the workflow) and imported in ProM. Hierarchical roles will be automatically flattened before simulation.
- YAWL enactment logs can be extracted from the engine and converted into the MXML format used by ProM. When analyzing the execution logs, we obtain execution time distributions for activities, a distribution of the case arrival rate, and value range distributions for boolean, numeric, and nominal data attributes.
- The information obtained from analyzing the enactment logs can be integrated with the imported YAWL model and the imported organizational model and directly used to automatically create a simulation model that can be loaded within CPN Tools. The conversion of the YAWL-based model to the Petri net-based model preserves all simulation-relevant information.
- While the generated CPN model can be used to perform the typical steady-state analysis, i.e., long-term behavior analysis, it also supports short-term simulation via loading a current execution state. The current state of the YAWL engine can be exported and re-loaded into the same CPN model at any point in time (i.e., without re-generating the model).
- Data attributes of all the basic types that can be chosen in the YAWL editor are imported. Variables of the YAWL type "boolean" are interpreted as the simulation model data type *boolean*, "double" and "long" as *numeric*, and the remaining types as *nominal*.
- If there is an initial or default value defined for a variable, then it is stored and used to initialize the data during simulation. If there is no initial or default value available, a random value will be generated upon initialization during simulation. This is useful for globally initialized data variables.

– When translating the YAWL model into a Petri net, implicit conditions need to be made explicit. Thereby, we adhere to the naming conventions of the YAWL engine to make it possible to load cases that are currently in a state involving such an implicit condition.

Limitations YAWL is a very rich workflow specification language, which, for example, provides the most extensive support in terms of Workflow Patterns compared to other systems. Since the described software is a proof-of-concept implementation of our approach, we had to make a few assumptions. They are described in the following.

– We do not support OR joins, multiple instances, or cancellation regions.
– The current CPN model only supports role-based allocation. It does not yet support rich resource allocation behavior supported by the YAWL engine. To be precise, we only read the *role* element within the *initialSet* of the *distributionSet* within the *offer* tag of the *resourcing* part. Furthermore, there should be not more than one role assigned to a task. This can be circumvented by defining "umbrella" roles that are defined as the union of existing roles.
– We assume that a particular resource is only working on at most one task (work item) at a time.
– Decision rules at OR splits are currently not imported.
– We do not yet support the import of data etc. from hierarchical YAWL models (sub nets to the root net will be seen as atomic tasks). However, the pure control-flow of hierarchical YAWL models can be imported and flattened in ProM.
– We assume that the (global) net variables and the corresponding (local) output parameters have the same name (which is not enforced by YAWL as explicit data mappings are established). Note that decisions related to link conditions are based on the net variables, and therefore can be simulated also on global attributes. However, the logged data variable changes carry the name of the local output parameter, and, therefore, cannot be automatically linked to the net variables imported with the YAWL model if they do not have the same name.
– YAWL allows for arbitrary data type extensions via XML Schema. This can of course not easily be handled with general purpose tools such as CPN Tools. Attributes that have a custom-defined data type are ignored.
– YAWL allows for arbitrary XPath expressions for data-based conditional routing. When importing the YAWL model, we currently only evaluate link conditions based on attribute and/or value comparisons via the operators $>$, $<$, $>=$, $<=$, $=$, and $!=$.

5 Discussion

In this paper we presented an innovative way to link workflow systems, simulation, and process mining. By combining these ingredients it becomes possible

to analyze and improve business processes in a consistent way. The approach is feasible, as demonstrated by our implementation using YAWL and ProM. To conclude, we would like to discuss the three main challenges that have been addressed in this research.

5.1 Faithful Simulation Models

Although the principle of simulation is easy to grasp, it takes time and expertise to build a good simulation model. In practice, simulation models are often flawed because of incorrect input data and a naïve representation of reality. In most simulation models it is assumed that resources are completely dedicated to the simulated processes and are eager to start working on newly arriving cases. In reality this is not the case and as a result the simulation model fails to capture the behavior of resources accurately. Moreover, in manually constructed models steps in the processes are often forgotten. Hence simulation models are usually too optimistic and describe a behavior quite different from reality. To compensate for this, artificial delays are added to the model to calibrate it and as a result its predictive value and trustworthiness are limited. In the context of workflow systems, this can be partly circumvented by using the workflow design (the process as it is enforced by the system) and historic data. *The approach presented in this paper allows for a direct coupling of the real process and the simulation model.* However, the generated CPN models in this paper can be improved by a better modeling of resource behavior. Moreover, the process mining techniques that extract characteristic properties of resources need to be improved to create truly faithful simulation models.

5.2 Short-term Simulation

Although most workflow management systems offer a simulation component, simulation is rarely used for operational decision making and process improvement. One of the reasons is the inability of traditional tools to capture the real process (see above). However, another, perhaps more important, reason is that existing simulation tools aim at strategic decisions. Existing simulation models start in an arbitrary initial state (without any cases in the pipeline) and then simulate the process for a long period to make statements about the steady-state behavior. However, this steady-state behavior does not exist (the environment of the process changes continuously) and is thus considered irrelevant by the manager. Moreover, the really interesting questions are related to the near future. Therefore, *the 'fast-forward button' provided by short-term simulation is a more useful option.* Because of the use of the current state and historic data, the predictions are more valuable, i.e., of higher quality and easier to interpret and apply. The approach and toolset presented in this paper allow for short-term simulation. In the current implementation the coupling between YAWL and ProM is not well-integrated, e.g., the translation of insights from simulation to concrete actions in the workflow system can be improved. Further research is needed to provide a seamless, but generic, integration.

5.3 Viewing Real and Simulated Processes in a Unified Manner

Both simulation tools and management information systems (e.g., BI tools) present information about processes. It is remarkable that, although both are typically used to analyze the same process, the results are presented in completely different ways using completely different tools. This may be explained by the fact that for a simulated process different data is available than for the real-world process. However, *the emergence of process mining techniques allows for a unification of both views*. Process mining can be used to extract much more detailed and dynamic data from processes than traditional data warehousing and business intelligence tools. Moreover, it is easy to extend simulation tools with the ability to record event data similar to the real-life process. Hence, process mining can be used to view both simulated and real processes. As a result, it is easier to both compare and to interpret 'what-if' scenarios.

Acknowledgements. This research was supported by the IOP program of the Dutch Ministry of Economic Affairs and by Australian Research Council grant DP0773012. The authors would like to especially thank Michael Adams, Eric Verbeek, Ronny Mans, and also Christian Günther, Minseok Song, Lindsay Bradford, and Chun Ouyang plus the review team for their valuable support in implementing the approach for YAWL and ProM. We also would like to thank Marlon Dumas for sharing his valuable insights during the many discussions we had about this topic.

References

1. W.M.P. van der Aalst, B.F. van Dongen, C.W. Günther, R.S. Mans, A.K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, and A.J.M.M. Weijters. ProM 4.0: Comprehensive Support for Real Process Analysis. In J. Kleijn and A. Yakovlev, editors, *Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007)*, volume 4546 of *Lecture Notes in Computer Science*, pages 484–494. Springer-Verlag, Berlin, 2007.
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, 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.
4. R. Ardhaldjian and M. Fahner. Using simulation in the business process reengineering effort. *Industrial engineering*, pages 60–61, July 1994.
5. J.A. Buzacott. Commonalities in Reengineered Business Processes: Models and Issues. *Management Science*, 42(5):768–782, 1996.
6. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
7. 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.

8. C. Hall and P. Harmon. A Detailed Analysis of Enterprise Architecture, Process Modeling, and Simulation Tools. Technical report 2.0, BPTrends, September 2006.
9. M. Jansen-Vullers and M. Netjes. Business process simulation – a tool survey. In *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, Aarhus, Denmark, October 2006. Published online at: http://www.daimi.au.dk/CPnets/workshop06/.
10. K. Jensen, L.M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *International Journal on Software Tools for Technology Transfer*, 9(3-4):213–254, 2007.
11. D.W. Kelton, R. Sadowski, and D. Sturrock. *Simulation with Arena*. McGraw-Hill, New York, 2003.
12. J. Kleijnen and W. van Groenendaal. *Simulation: a statistical perspective*. John Wiley and Sons, New York, 1992.
13. M. Laugna and J. Marklund. *Business Process Modeling, Simulation, and Design*. Prentice Hall, Upper Saddle River, New Jersey, 2005.
14. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
15. H. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*, volume 2617 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2003.
16. H.A. Reijers and W.M.P. van der Aalst. Short-Term Simulation: Bridging the Gap between Operational Control and Strategic Decision Making. In M.H. Hamza, editor, *Proceedings of the IASTED International Conference on Modelling and Simulation*, pages 417–421. IASTED/Acta Press, Anaheim, USA, 1999.
17. S.M. Ross. *A course in simulation*. Macmillan, New York, 1990.
18. A. Rozinat, R.S. Mans, M. Song, and W.M.P. van der Aalst. Discovering Colored Petri Nets From Event Logs. *International Journal on Software Tools for Technology Transfer*, 10(1):57–74, 2008.
19. Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, Berlin, Heidelberg, 2007.
20. M.T. Wynn, M. Dumas, C.J. Fidge, A.H.M. ter Hofstede, and W.M.P. van der Aalst. Business Process Simulation for Operational Decision Support. In A.H.M. ter Hofstede, B. Benatallah, and H.-Y. Paik, editors, *BPM 2007 Workshops*, volume 4928 of *Lecture Notes in Computer Science*, pages 66–77. Springer-Verlag, 2008.