Business Process Simulation for Operational Decision Support

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

[1] Faculty of Information Technology, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001, Australia.
{*m.wynn,m.dumas,c.fidge,a.terhofstede*}*@qut.edu.au*
[2] Dept. of Mathematics and Computer Science, Eindhoven University of Technology, PO Box 513, NL-5600 MB Eindhoven, The Netherlands.
w.m.p.v.d.aalst@tue.nl

Abstract. Contemporary business process simulation environments are geared towards design-time analysis, rather than operational decision support over already deployed and running processes. In particular, simulation experiments in existing process simulation environments start from an empty execution state. We investigate the requirements for a process simulation environment that allows simulation experiments to start from an intermediate execution state. We propose an architecture addressing these requirements and demonstrate it through a case study conducted using the YAWL workflow engine and CPN simulation tools.

1 Introduction

Business process simulation enables the analysis of business process models with respect to performance metrics such as throughput time, cost or resource utilization. A number of business process modelling tools support simulation to varying degrees [7]. However, this tool support is largely geared towards *a priori*, i.e., design time, comparison of candidate business process models. Accordingly, they assume that simulation experiments are run from an empty initial state, for a very large number of cases, to give analysts insight into the average, long-term benefits of process improvement options.

This contrasts markedly with the requirements of *operational decision support*, where the goal is to evaluate short-term options for adjusting an already deployed business process in response to contextual changes or unforeseen circumstances. In this situation, the current system state and recent event history cannot be ignored, and the emphasis is on understanding the short-term implications of making a change to the system.

Another shortcoming of contemporary process simulation tools with respect to operational decision support is the inability to set different completion horizons for simulation experiments. The focus of traditional simulation experiments is to identify average long-term behaviour, over a wide variety of contextual scenarios. By contrast, operational decision making introduces the need to make

short-term decisions, based on the current state and specific recent history. To do this we need the ability to limit the simulation's forward-looking 'horizon', to enable rapid evaluation of the consequences of various decisions. A typical example is the need to determine if redeploying resources will eliminate a temporary backlog of unprocessed jobs within a given time frame. Simulation horizons of interest include absolute times (e.g., 30 June 9pm), time durations (e.g., 5 hours from now), the number of jobs completed (e.g., 200^{th} case), and the number of resources consumed (e.g., when 80% of employees are busy).

In this paper we define the requirements for an operational process simulation environment which addresses these issues, and describe a suitable toolset architecture. To demonstrate the feasibility of the concept, we also describe the outcomes of a proof-of-concept case study performed using existing, off-the-shelf tools, the YAWL workflow engine and the CPN simulation tools.

2 Previous and related work

Business process simulation involves developing an accurate simulation model which reflects the behaviour of a process, including the data and resource perspectives, and then performing simulation experiments to better understand the effects of running that process [13]. In general, a business process simulation model consists of three components: basic model building blocks (e.g., entities, resources, activities, and connectors); activity modelling constructs (e.g., branch, assemble, batch, gate, split and join); and advanced modelling functions (e.g., attributes, expressions, resource schedules, interruptions, user defined distributions) [13]. Business process simulation is regarded as an invaluable tool for process modelling due to its ability to perform quantitative modelling (e.g., cost-benefit analysis and feasibility of alternative designs) as well as stochastic modelling (e.g., external factors and sensitivity analysis) [4]. Simulation has been used for the analysis and design of systems in different application areas [13], a "decision support tool" for business process reengineering [6] and for improving orchestration of supply chain business processes [12].

Simulation functionality is provided by many business process modelling tools based on notations such as EPCs or BPMN. These tools offer user interfaces to specify basic simulation parameters such as arrival rate, task execution time, cost, and resource availability. They allow users to run simulation experiments and to extract statistical results such as average cycle time and total cost. Process simulation can also be performed using a more general class of simulation techniques known as discrete event simulation [13].

Even though simulation is well-known for its ability to assist in long-term planning and strategic decision making, it has not to date been considered a mainstream technique for operational decision making due to the difficulty of obtaining real-time data in the timely manner to set up the simulation experiments [8]. Nevertheless, a number for recent developments point out how aspects of the problem can be handled, and form the basis of our approach.

A novel use of discrete event simulation, close to our own aims, is short-interval scheduling of a shop floor control system where the ability of the simulation to "look ahead" at the expected performance of the system in the near future, given its current status, is used to provide real-time responses to dynamic status changes [3, 11]. We aim to generalise this specific capability to arbitrary business models. More significantly, Reijers et al. [8], introduced the concept of 'short-term simulation'. They went on to experiment with short-term simulations from a 'current' system state to analyse the transient behaviour of the system, rather than its steady-state behaviour [9]. A similar resource-oriented approach is provided by the proprietary Staffware prediction engine[1]. Our goal is to design such a 'short-term' analysis architecture in the context of widely-used, off-the-shelf workflow tools, and without the specific focus on resourcing.

To do this, we have experimented with a combination of the YAWL workflow engine [1] and the CPN Tools simulator [2]. A number of previous such experiments have informed our work. For instance, Gottschalk et al. [5] used a YAWL subset to generate CPN models, and Verbeek et al. [14], integrated the ExSpect simulator with Protos 7.0 to provide modelling and simulation facilities in one tool. Also, Rozinat et al. [10] showed how event logs produced by CPN models can be 'mined' to discover the operational characteristics of the model. Our aim is to combine both these notions, i.e., creating simulation models from workflow processes and feeding back simulation results to calibrate the model, but with a particular emphasis on incorporating observed behaviours from the 'real', operational system into the predictive simulation.

3 Requirements for operational process simulation

In this section we use a simple example to motivate the requirements for operational decision support. Consider the credit card application process expressed as a workflow model in Figure 1. The process starts when an applicant submits an application. When an application is received, a credit clerk checks whether the application is complete. If the application is found to be incomplete, the clerk requests additional information and waits until the information is received before proceeding. For a complete application, the clerk performs further checks to validate the applicant's income and credit history. The validated application is then passed on to a manager to make the decision. The manager decides either to accept or reject an application. For an accepted application, a credit card is produced and delivered to the applicant. For a rejected application, the applicant is given a timeframe to request a review of the decision. If a review request is not received, the process ends.

A typical question for the credit application process might be "How long will it take to process a credit card application?" Using conventional tools for business process simulation, it is possible to answer this question with an *average* duration, assuming some 'typical' knowledge regarding the available resources

[1] `http://www.tibco.com/`

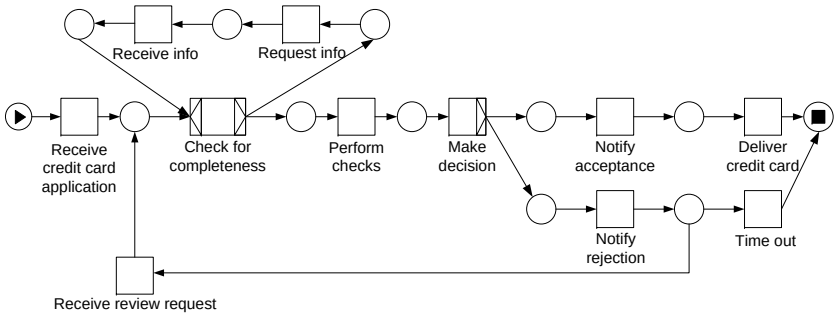

Fig. 1. Workflow model of a credit card application process

and expected execution times for the involved activities. However, if the business process is already operational, and it is supported by a workflow management system, the same question can be asked for observed, specific states of execution. For instance, we could ask ourselves, "how long will it take to complete processing a particular application, provided that all documentation is complete and the application is now ready for a manager to make the decision?" Most importantly, this can be done using the actual state of the system's resources, such as the number of clerks already occupied with other applications.

While performing 'short-term' system predictions, we need to define when a simulation experiment should stop, i.e., the completion horizon. This can be defined as a bound on various aspects of simulation, such as end times and durations, as well as the number of case completions and at various resource utilisation rates. For the credit card application example, some interesting completion horizons include: 12 or 24 hours duration from now; the time at which the delay for decision making is over 3 days; the point at which 1000 applications have been processed, etc. Operational decision makers seeking to adjust the credit card process following spikes in demand, or delays caused by unexpected events, would benefit from being able to perform simulations with different horizons.

Consider for example the case where the company runs a highly successful promotion campaign and receives unexpectedly large numbers of credit applications. As a result, the company now has a backlog of applications (e.g., 100 applications) waiting to be processed. In this case, the average time (e.g., five days) to process a credit card application cannot be guaranteed with the current number of staff members (3 clerks and 1 manager). At this stage, it is desirable to obtain more realistic input data to determine the cycle time by taking into consideration the current number of applications in the queue, and other observable properties of the 'live' system. Understanding this can lead to a more effective resource planning for the manager. Given the current state of system, the following questions might be of interest to a manager:

1. What is the cycle time to process an application at this current load?
2. Is it possible for all applications in the queue to be processed after a certain duration (e.g., in 12 hours)?

3. What are the consequences of adding five additional clerks and two managers to assist in processing?

None of these questions can be answered with precision using the 'average' results produced by a conventional simulation from an empty state. Overall, therefore, the requirements for an operational process simulation toolkit are:

1. The ability to start a simulation from a non-empty state, using data obtained from the operational system's actual behaviour.
2. The ability to specify (multiple) breakpoints in a simulation experiment based on different criteria such as the number of cases completed, the time horizon, or based on conditions encountered in the simulated environment (e.g., queue or resource utilization dropping below preset levels).
3. The ability to automatically extract and process historical execution data, and in particular recent data, in order to calibrate the simulation model.

4 Architecture for operational process simulation

In this section, we first propose the generic architecture for simulation and then discuss the various process components to realise this architecture.

4.1 Generic architecture

Figure 2 shows a data flow diagram of the proposed architecture as a tool chain to support operational process simulation. The process modelling and analytics phase of the tool chain is concerned with developing a stateful simulation model while the process simulation phase focuses on running various simulation experiments and providing simulation reports as well as detailed simulation logs for use as input into (re)design of the simulation model. The diagram shows a "step-by-step" translation of a simulation template: first by enriching the template with historical data to derive the various simulation parameters and second by including the starting state to develop a stateful simulation model. The resulting stateful simulation model is then used to run various simulation experiments. The external input from observed "real-world" logs plays a crucial role in this architecture and it is envisioned that a number of extraction functions will be used to derive the historical data and the starting state from these logs. The architecture also supports the use (and conversion) of simulation logs to derive historical data and a starting state.

The main data objects (depicted as hexagons) and activities (depicted as rectangles) comprising the architecture in Fig. 2 are as follows.

Simulation template A simulation template includes the representation of control, data, and resource requirements of a business process (process definition) as well as necessary setup information for simulation experiments. To run simulation experiments, a simulation template defines various input and output parameters, breakpoints for completion horizons and derivation functions. At

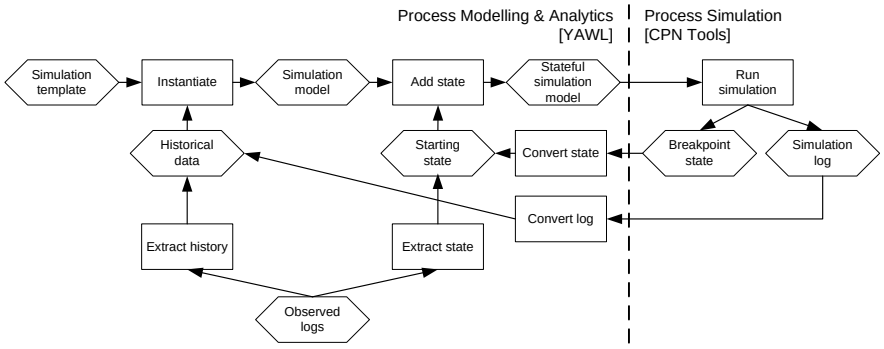

Fig. 2. Architecture of the operational process simulation toolkit

a minimum, a simulation template needs to specify the following setup information: arrival rates of cases, a resource calender, simulation parameters (Key Performace Indicators), completion horizons (breakpoints) and simulation report requirements (monitors). Furthermore, various parameters in the template are also enriched with information on how to generate the data used in the simulation (estimated or derived). In our proposed architecture such parameters can be specified either by entering estimations or by specifying various derivation functions over the observed and simulated log files. For example, the case arrival rate parameter is typically specified over a Poisson distribution. Similarly, the average execution time of a task is specified using mean and standard deviation. Abstract data types for case arrival rates and execution times in a simulation template can be specified as follows:

$$ArrivalRate : (ArrivalRateFunction \cup HistoricalData \rightarrow ArrivalRateFunction)$$
$$ExecutionTime : Task \rightarrow (TimeFunction \cup HistoricalData \rightarrow TimeFunction)$$

Instantiation This activity takes a simulation template with derivation functions and historical data from the logs to generate a simulation model. It is essential that log data contains relevant information that can be used for a given derived parameter. Obviously, the requirements for logs could vary depending on a given parameter and the derivation function used. The logs data can be based either on observations from a running process engine or from prior simulation logs.

Historical data Historical data to instantiate a simulation template could be extracted from the execution logs of a process engine or from previous simulation runs. For instance, if an average execution time of a task is to be derived using log data to calculate the mean and standard deviation, the log should contain information about when all instances of a given task are executed and completed. If a derivation function is also based on resources (i.e., the time it takes to execute a task by a manager), then the log should contain information about resource utilisation in addition to time. Conversions and adjustments might be necessary if log data is incompatible with the requirements in the template.

These conversions take place during the *Extract history* activity for observed logs and the *Convert log* activity for simulation logs. The necessary abstract data type for a log to derive case arrival rate and execution times is as follows:

$$Cases : Case \times CreationTime \times CancellationTime \times CompletionTime$$
$$CompletedTasks : Case \times Task \times StartTime \times EndTime \times Resource$$

Observed logs and Simulation logs While the observed logs represent the data and metrics from executing process engines, the simulation logs provide the information from prior simulation runs. Both types of historical data are useful in determining appropriate values for simulation parameters.

Simulation model After ensuring that all derivable simulation parameters have been instantiated with historical data, a simulation model is generated. It is now possible to use this simulation model to run simulations from the initial state.

Adding a state This activity takes a simulation model and a given state to set the starting state of a simulation experiment. If a simulation experiment is to be started from scratch (an empty state), minimal transformation is required to include resource scenario for a simulation experiment. On the other hand, if a simulation experiment is to be started from a given state, current state information is added to obtain a stateful simulation model. In cases where some of the tasks are already running for a certain amount of time in the simulation's starting state, we propose to use a truncated probability distribution so that the duration randomly assigned to an active task during the simulation is always greater than the amount of time for which the task has already been running.

Starting state The state information can be derived from historical logs and also from prior simulation runs. At a minimum, the logs used to derive state should contain information on active cases, resource availability and active and enabled tasks information. Inconsistencies are possible between a given model and the data obtained from the logs and conversions might be necessary. The following abstract datatypes for logs capture the minimum information requirements to generate an initial simulation state:

$$ActiveCases : Case \times CreationTime$$
$$ActiveTasks : Case \times Task \times StartTime \times Resource$$
$$EnabledTasks : Case \times Task$$
$$ResourceAvailability : Resource \times Role$$
$$LogTime : Time$$

Importantly, this allows us to inject observed characteristics of the system into the simulation. For instance, let's assume a simulation model may specify the (initial) availability of three staff members, whereas the observed logs show there are actually five staff members currently assigned to this process.

Breakpoint state Capturing the full state of the simulation model at the end of a simulation experiment provides an opportunity to use the breakpoint state as the initial state for another simulation, thus facilitating the conduct of simulation experiments with different breakpoints.

Stateful simulation model A stateful model is obtained by enriching a simulation model with starting state information for simulation runs. More than one stateful simulation model can be developed where each one represents the state at a certain point in time. The *LogTime* parameter from the state is used to set the starting time of the simulation experiments.

Running the simulation Simulation experiments can now be started using a stateful model and stopped at various completion horizons. In addition to the generation of simulation reports for analysis, the architecture makes provision for the generation of both breakpoint states and simulation logs. This data can then be used as input for later simulation runs after necessary conversions.

4.2 A practical instantiation of the architecture

It is possible to realise the proposed simulation architecture in a number of ways using a suitable process editor, a process engine (with logging functionality) and a simulation tool that is flexible enough to support our requirements. For our research, we are using the YAWL workflow environment for both modelling and analytics components and the simulation capabilities within CPN Tools for process simulation, as shown by the partition in Fig. 2.

The YAWL workflow environment was chosen because of its formal foundation in Petri nets, its expressiveness in providing support for workflow patterns, its easy-to-use graphical editor that has the ability to generate executable process models, and its extensive logging function for process execution. There are also mappings available between various business process modelling notations (EPC, BPMN, BPEL) and Petri nets. Furthermore, the YAWL workflow language is supported by an open-source implementation[2]. The YAWL editor is an ideal candidate for the process modelling component in the architecture as the user can specify control, data and resource requirements of a business process using a natural graphical notation and then export the process definition as an XML file ready for execution in the engine. Various verification functionalities are also available in the editor to ensure the correctness of the process model before execution. The YAWL editor can be easily extended to capture various setup parameters for a simulation template. The logging module in the YAWL engine can be used to record the statistics of various cases (such as the start and end times, the resource, whether the task is cancelled or completed, etc). These logs provide sufficient information to generate stateful simulation models.

The current YAWL implementation does not provide simulation functionality. However, it is rather straightforward to transform YAWL models into

[2] `http://www.sourgeforge.net/yawl`

Coloured Petri Nets (CPNs) [2], modulo some restrictions, and to exploit the simulation capabilities of CPN Tools[3]. Coloured Petri nets can be used to model Petri nets with time constraints and hierarchy. In contrast to contemporary BPM simulation tools, CPN Tools can be customised to support our requirements, i.e., the ability to start simulation experiments from a given state and the specification of different completion horizons using breakpoints. It is possible to incorporate the log data by specifying derivation rules using ML functions. Rather than modelling the business process directly in CPN Tools (i.e., modelling the process as one or more Coloured Petri nets), our arrangement avoids the need for detailed knowledge of Petri nets, which would make CPN Tools unsuitable for business process designers. For these reasons, we combine use of YAWL for modelling and execution of business processes with CPN Tools for BPM simulation, to leverage their strengths in their respective areas.

5 Proof of concept

To validate the simulation architecture, we developed a stateful simulation model for the credit card application process from Section 3 and performed simulation experiments using the CPN Tools. The credit card application workflow process from Fig. 1 was (manually) translated into a corresponding hierarchical and timed colored Petri net as depicted in Fig. 3.

The translations are similar to the Gottschalk et al.'s approach [5] where a YAWL condition is mapped to a CPN place and a YAWL task to a CPN substitution transition (see the *Make decision* task Fig. 4). As the credit card process does not contain tasks with complex split and join behaviours, we elected to map most YAWL tasks to CPN transitions directly for simplicity. The *Environment* subpage controls the arrival of cases. Tokens in the *resource* place indicates the number of employees available to execute the tasks (e.g., clerks and managers). Execution times for tasks were specified using ML functions and a time delay was attached to the task (e.g.,$a@+getExecTime(mean = 7200, stdD = 3600)$). In Fig. 4, the XOR-split behaviour of the task is modelled as two transitions that share the same input place (*busy*). The start transition for the *Make decision* task has another input from the resource place with a guard to that only a manager can carry out this task ($[\#role(e) = mgr]$).

To determine the feasibility of the proposed architecture, we tested out the resource planning scenario as discussed in section 3. For testing purposes, the stateful simulation model is populated with the following data.

- a case arrival rate of 1 application per hour following a Poisson distribution;
- 100 active cases as the starting state where 80 credit card applications are in the *ready* place, 15 in the *complete check* place, 3 in the *receive* place, 1 in the *accept* place and 1 in the *busy* place of the *Make decision* task;
- breakpoint monitors with 12-hour and 24-hour time horizons and marking size monitors to observe the two places of interest, the *ready* place and the *complete check* place;

[3] `http://www.daimi.au.dk/CPnets/`

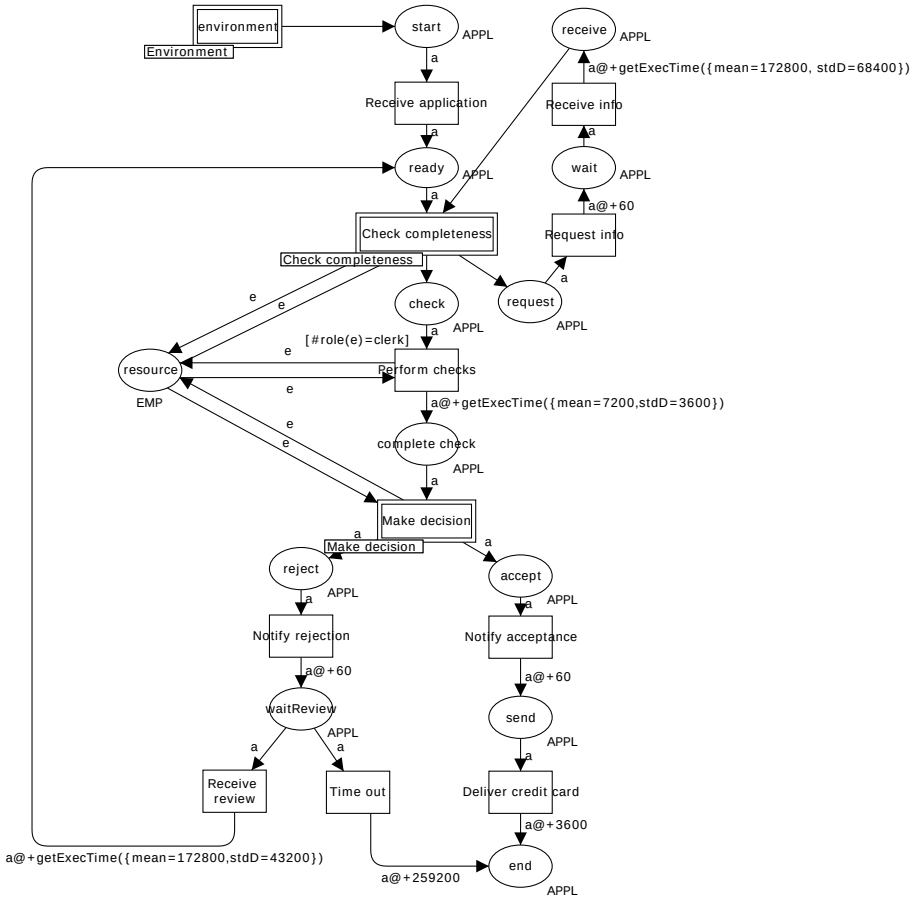

Fig. 3. CPN model of credit card application process

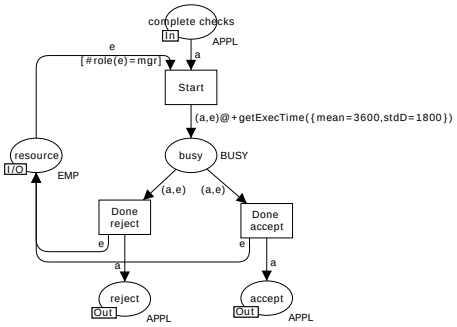

Fig. 4. CPN subpage for making a credit card decision

– execution times (sample mean and standard deviation) for each task; and
– two resource availability scenarios, namely 5 clerks and 1 manager, and 10 clerks and 3 managers.

Multiple simulation runs are carried out using the starting state with 100 active cases, two different time horizons (12 hours and 24 hours) and with two different resource scenarios. The results (see Table 1) show that when processing the applications with 5 clerks and 1 manager, on average (with 95% probability) between 48 to 55 applications are still in the queue in the *ready* (R) place and between 17 to 28 in the *complete check* (C) after 12 hours and it becomes 13-19 (R) and 14-22 (C) after 2 24 hours. On the other hand, the queue is reduced to 12-20(R) and 3-9 (C) after 12 hours, and 10-17(R) and 0-2(C) when 10 clerks and 3 managers are available. This scenario illustrates the possibilities opened by operational process simulation, in terms of being able to perform "what-if" analysis based on the current situation and to different completion horizons.

Table 1. Summary of simulation results

Resource availability	Duration (12 hours)	Duration (24 hours)
5 clerks and 1 manager	48-55 (R) and 13-19 (C)	17-28 (R) and 14-22 (C)
10 clerks and 3 managers	12-20 (R) and 3-9 (C)	10-17(R) and 0-2 (C)

In this proof-of-concept demonstration we were specifically interested in validating the feasibility of inserting a non-empty starting state and multiple completion horizons into simulation experiments. For instance, we assumed an arrival rate of 1 application per hour instead of deriving the actual arrival rate for these observations. Nevertheless, we have confirmed that the YAWL engine logs contain sufficient data to instantiate the simulation template and to add state. The same goes for the resource availability statistics and the current state. In the next stage, we plan to implement the necessary extraction and conversion functions to derive simulation parameters and starting states automatically from logs.

6 Conclusion and future work

To produce accurate short-term predictions a workflow simulation environment must start its analysis in a state that incorporates the actual, observed properties of the operational system, including its recent history. In this paper we have demonstrated the feasibility of building such a simulation environment using off-the-shelf workflow modelling and system simulation tools. A general design for such a system was defined in terms of its essential capabilities and a (manual) feasibility study was conducted using the YAWL and CPN Tools toolkits. Currently we are implementing the various 'gluing' components needed to automate the transformation of 'mined' log values to produce simulation inputs.

Acknowledgement This research was funded by the Australian Research Council grant DP0773012,

References

1. W.M.P. van der Aalst and A.H.M ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, June 2005.
2. M. Beaudouin-Lafon, W. Mackay, P. Andersen, P. Janecek, M. Jensen, H. Lassen, K.Lund, K.Mortensen, S. Munck, A. Ratzer, K. Ravn, S. Christensen, and K. Jensen. A Tool for Editing and Simulating Coloured Petri Nets. In *ETAPS 2001*, volume 2031 of *LNCS*, pages 574–577, Genova, Italy, April 2001. Springer-Verlag.
3. G. Drake, J. Smith, and B. Peters. Simulation as a planning and scheduling tool for flexible manufacturing systems. In *Proceedings of the 27th conference on Winter simulation*, pages 805–812, 1995.
4. G. Giaglis, R. Paul, and G. Doukidis. Simulation for intra- and inter-organisational business process modelling. In *Proceedings of the 28th conference on Winter simulation*, pages 1297–1308, 1996.
5. F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, and H.M.V. Verbeek. Protos2CPN: Using Colored Petri Nets for Configuring and Testing Business Processes. 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/`.
6. V. Hlupic and S. Robinson. Business process modelling and analysis using discrete-event simulation. In *Proceedings of conference on Winter simulation*, volume 2, pages 1363–1369, 1998.
7. 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/`.
8. H.A. Reijers and W.M.P. van der Aalst. Short-term simulation: bridging the gap between operational control and strategic decision making. In *Proceedings of the IASTED Conference on Modeling and Simulation*, pages 417–421, Philadelphia, USA, 1999.
9. Hajo A. Reijers. *Design and Control of Workflow Processes*. Springer-Verlag New York, Inc., Secaucus, USA, 2003.
10. A. Rozinat, R. Mans, and W.M.P. van der Aalst. Mining CPN Models: Discovering Process Models with Data from Event Logs. 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/`.
11. J. Smith, R. Wysk, D. Sturrock, S. Ramaswamy, G. Smith, and S. Joshi. Discrete event simulation for shop floor control. In *Proceedings of conference on Winter simulation*, volume 2, pages 962–969, Lake Buena Vista, FL, December 1994.
12. T.W. Tewoldeberhan, A. Verbraeck, and S. Msanjila. Simulating process orchestrations in business networks: a case using BPEL4WS. In *Proceedings of the 7th international conference on Electronic commerce*, pages 471–477, New York, USA, 2005. ACM Press.
13. Kerim Tumay. Business process simulation. In *Proceedings of the 28th conference on Winter simulation*, pages 93–98, 1996.
14. H.M.V Verbeek, M. van Hattem, H. Reijers, and W. Munk. Protos 7.0: Simulation made accessible. In *Proceedings of ATPN 2005*, volume 3536 of *LNCS*, pages 465–474. Springer, 2005.