Generating Event Logs with Workload-Dependent Speeds from Simulation Models*

Joyce Nakatumba, Michael Westergaard, and Wil M.P. van der Aalst

Eindhoven University of Technology, The Netherlands
{jnakatum,m.westergaard,w.m.p.v.d.aalst}@tue.nl

Abstract. Both simulation and process mining can be used to analyze operational business processes. Simulation is model-driven and very useful because different scenarios can be explored by changing the model's parameters. Process mining is driven by event data. This allows detailed analysis of the observed behavior showing actual bottlenecks, deviations, and other performance-related problems. Both techniques tend to focus on the control-flow and do not analyze resource behavior in a detailed manner. In this paper, we focus on *workload-dependent processing speeds* because of the well-known phenomenon that people perform best at a certain stress level. For example, the "Yerkes-Dodson Law of Arousal" states that people will take more time to execute an activity if there is little work to do. This paper shows how workload-dependent processing speeds can be incorporated in a simulation model and learned from event logs. We also show how event logs with workload-dependent behavior can be generated through simulation. Experiments show that it is crucial to incorporate such phenomena. Moreover, we advocate an amalgamation of simulation and process mining techniques to better understand, model, and improve real-life business processes.

1 Introduction

Process mining is a technique that extracts knowledge from *event logs* recorded by information systems [2]. Most systems are able to *sequentially* record *events* such that each event refers to an *activity* (i.e., a well-defined step in the process) and is related to a particular *case* (i.e., a process instance). The event logs of such systems also store more information about events, for example, the *resource* (i.e., person or device) executing or initiating the activity, and the *timestamp* of the event. Examples are Enterprise Resource Planning systems (SAP, Oracle), Business Process Management Systems (Hospital Information Systems) Using process mining techniques it is possible to discover processes from event logs. Moreover, event logs can be checked to assess conformance with respect

* This research is supported by the Technology Foundation STW, applied science division of NWO and the technology program of the Dutch Ministry of Economic Affairs.

M. Bajec and J. Eder (Eds.): CAiSE 2012 Workshops, LNBIP 112, pp. 383–397, 2012.

to defined processes and process models can be modified and extended using process mining techniques. This provides necessary insights to manage, control, and improve business processes [2]. In many of the business processes supported by information systems, human resources are the limiting factor, i.e., delays are often caused by the unavailability or overloading of people. Understanding such delays is vital for process improvement. Therefore, we focus on the analysis of event logs where most activities are executed by human resources.

Simulation is an important technique that can be used to, for example, analyze the performance of business processes, verify that a business process will work as designed, gain insights into the effects of particular business decisions. All of these tools use information about tasks, resources, and the ordering of tasks to calculate various performance indicators. Whereas simulation is a well established area of computing, there are several challenges when mapping a real life business process onto a simulation model especially when involving humans. Several pitfalls of current simulation models have been discussed in [1, 3]. One of these is that the resource models built in simulations are usually very simple and are not a true reflection of reality. However, in this paper we deal with the modeling of resources in simulation models by adding additional parameters to the resource model in order to make sure that the process is modeled accurately. We focus on the problem of the *effect of workload on processing speeds*.

We provide a refined view on modeling resource behavior by taking into consideration the fact that the processing speeds of resources are not merely dependent on the service rates of the tasks but also on the workload present in the system. This is based on various studies in the area of psychology and operations research that suggest a relation between workload and performance of workers [5, 13, 16]. Such behavior is typical in organizations and in this paper, we build a simulation model based on colored Petri nets (CPNs) [6] that models workload influence on processing speeds of resources. Further on, we carry out experiments and the aim is to compare the simulated resource performance while taking workload into account with a simulated resource performance without the workload effect. Our experimental results indeed show that workload does have an effect on resource performance and should not be ignored when building simulation models. However, to adequately set the resource behavior parameters in simulation models, we can also exploit information available in event logs using process mining techniques. Given an event log, we can learn information characterizing workload-dependent speeds using process mining. This approach is implemented in the process mining framework ProM [14]. Moreover, we see the same relationship modeled in a simulation model also exists in real life based on the ProM analysis.

Accurate modeling of resource behavior in simulation models is important for two main reasons. First, a better modeling of resource behavior in simulation models will help make simulation models that are more realistic and also tightly coupled to Process-aware Information Systems [1, 15]. Using simulations, we aim at the *generation and extension of event logs with workload-dependent speeds*. Log generation is a very important aspect because we are able to change model

parameters or include different process alternatives in the simulation model, run simulations while obtaining new event logs. The generated event logs can be analyzed using process mining techniques to verify the effects of the parameter changes on resource behavior. This is not only relevant for analyzing alternatives for a concrete process, but also for the evaluation of process mining techniques focusing on the resource and time perspectives. Hence, event logs from the simulated environment and the real world can be compared using the same technique, i.e., process mining as shown in Fig. 1.

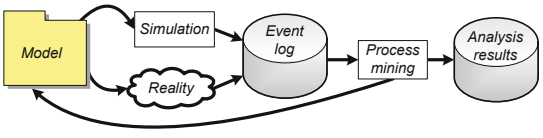

Fig. 1. Process mining is able to analyse event logs from different sources, i.e., Simulation and Reality

Second, the information about modeling resource behavior in simulation models can be used as a basis for *operational support and providing of more realistic simulations of resources*. For example, in [12] we discuss a testing platform for simulating resource behavior where users interacting with a workflow system are provided with on-line information about running processes and recommendations about the next actions to take in order to arrive at a goal [1]. Here, we provide for more realistic modeling of resource behavior in simulation models while taking workload present in the system into account.

The remainder of the paper is organized as follows. First, we provide an overview of workload-dependent speeds in Section 2 and discuss the approach taken to model workload-dependent speeds in a simulation model. In Section 3, we discuss the CPN model developed for modeling the effect of workload on processing speeds. In Section 4, we discuss the experiments carried out based on the simulation model. Section 5 has a discussion of related work. Section 6 concludes the paper.

2 Background

In [1, 3], we discussed several problems that can arise when building simulation models involving human resources. For example, when resources are working they tend to distribute their attention over multiple processes, they also work part-time and in batches. However, when resources are working it is not only their distribution of attention over different processes that can affect their performance. In this paper, we argue that the absolute working speeds of resources also determine their capacity for a particular process.

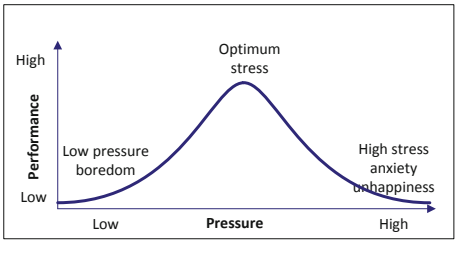

Fig. 2. Yerkes-Dodson Law modeled as U-shaped Curve

The speed at which resources work is in many systems partly determined by the amount of work that is currently present [13]. For example, in busy

periods of the year people tend to increase their speed to process more cases. However, when people are given too much work over a long period of time, their performance then tends to drop. In the literature, this phenomenon is known as the "Yerkes-Dodson Law of Arousal" [16]. This law models human performance as an inverted U-shaped curve as depicted in Fig. 2. If the law holds, the performance of people (i.e., the speed at which they work) is determined by the workload that is currently present in the system [5, 11, 13]. This implies that for a given individual and a given set of tasks, there is an optimal level at which the performance of that individual has a maximal value and beyond this optimal level, the worker's performance collapses. In this paper, we do not try to represent this very complex world in a simulation model. However, we use simple parameters depending on the work-items present in the system. We limit our focus to a single process involving multiple resources and tasks and represent this in a simulation model.

2.1 Approach

When resources work, they execute a number of work-items for each case. In Fig. 3, we give an example of how work-items can be handled by two resources r_1 and r_2 over a 12-hour time period. The horizontal axis indicates the time over which work-items are executed while the vertical axis indicates the order in which the work-items are executed. Each box shown in the figure corresponds to a work-item executed by a resource. The start of the box indicates when a work-item is scheduled, the start of the shaded part of the box indicates when the work-item is started and the end of the box shows when the work-item is completed. For example, r_1 executes work-items w_1-w_9. and starts with the execution of w_1, followed by w_2 etc. From Fig. 3, we observe work-item executions at different points in time for each resource, i.e., point p_1 for r_1. If we consider point p_2 corresponding to resource r_2 we observe that:

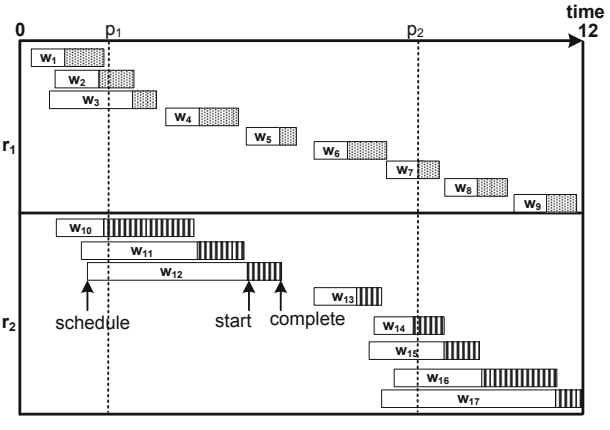

Fig. 3. Resources r_1 and r_2 execute work-items over a 12 hour period

- in the *allocated* (i.e., scheduled but not started) state, there are work-items w_{15}, w_{16} and w_{17},
- in the *executing* state, there is work-item w_{14}, and
- in the *finished* state, there are work-items w_{10}, w_{11}, w_{12} and w_{13}.

We have different properties of a work-item for example, when it was scheduled, started and completed, and we can use these to define further work-item properties that can be used to define workload.

Definition 1 (Work Item Properties). Let $\mathcal{R}$ be a set of resources, $\mathcal{T}$ be the time domain and $\mathcal{W}$ be a set of work-items. For any **work-item** $w \in \mathcal{W}$, we define the following **properties**:

- $t_{sch}(w) \in \mathcal{T}$ *as the schedule time of* w,
- $t_s(w) \in \mathcal{T}$ *as the start time of* w,
- $t_c(w) \in \mathcal{T}$ *as the completion time of* w,
- $r(w) \in \mathcal{R}$ *as the resource scheduled to execute* w.

From the three work-item states, we can define metrics for calculating workload. Although many workload definitions are possible [5,13], in this section we define workload based on the two main perspectives.

i. The *Queue Length* perspective specifies the amount of work scheduled for a given resource, and
ii. The *How Busy* perspective specifies the amount of work that each resource has executed in the recent past.

During the simulation it is possible to look at all the work-items that have been scheduled for a given resource before they start execution of a particular work-item. We now formally define workload based on the queue length perspective.

Definition 2 (Workload from Queue Length Perspective). Let $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{W}$ be as defined in Def. 1. We define workload based on **queue length** perspective for a resource $r \in \mathcal{R}$ at a specific time $t \in \mathcal{T}$ by function $ql : \mathcal{R} \times \mathcal{T} \rightarrow \mathbf{N}$ where $\mathbf{N}$ is the set of natural numbers as: $ql(r,t) = |\{w \in \mathcal{W} \mid t_{sch}(w) \leq t \leq t_s(w) \wedge r(w) = r\}|$

Hence function ql counts the number of work-items that have been scheduled for a resource, however, at the point at which we measure workload, the work-items have been not been started. This can be computed on-the-fly by counting when a work-item is scheduled and this value can be decremented when the work-item is started.

We also define how to calculate workload based on the how busy perspective. However, in the how busy perspective we need to define a *horizon period* over which we can measure the number of completed work-items.We now formally define workload based on the queue length perspective.

Definition 3 (Workload from How Busy Perspective). Let $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{W}$ be be as defined in Def. 1. Given a specific horizon period $h \in \mathcal{T}$, we define workload based on the **how busy** perspective for a resource $r \in \mathcal{R}$ at time $t \in \mathcal{T}$ by function $bu : \mathcal{R} \times \mathcal{T} \rightarrow \mathbf{N}$ as: $bu(r,t) = |\{w \in \mathcal{W} \mid t - h \leq t_c(w) \leq t \wedge r(w) = r\}|$

Here, we consider all the work-items that a resource has executed in a specific time period. Hence, given the current time t, we can define a time period for example, $t - h$ where h is a specific time period. If h is defined as 10 (cf. Fig. 3), this implies that we look at all the work-items that were completed no more than 10 time units before the resource starts execution of the current work-item. The schedule moment of a work-item can be derived as the point in time when a case arrives in the model. In this model, we are interested in the effect of workload on the processing speeds of resources. Although we here measure

workload given a specific horizon period, it is also possible to use a function for assigning various weights to work-items depending on how long ago they were executed, assigning higher weight to items completed the same day but lower weight to items completed last week.

We characterize processing speeds as the flow time associated with each case. The flow time is the total time the case spends in the system, i.e., the difference between when the case is created and when the last task for the case is completed.

Definition 4 (Processing Speeds). Let $t_{sch}(w)$, $t_s(w)$, and $t_c(w)$, be as defined in Def. 1. We define the **service time** and **flow time** attached to each work item as:

- $st(w) = t_c(w)$ - $t_s(w)$ is the service time associated to w, and
- $ft(w) = t_c(w)$ - $t_{sch}(w)$ is the flow time associated to w.

The main parameters of the simulation model are as follows (a) *arrival rate* λ ($\lambda > 0$) the average number of cases arriving per time unit, (b) *service rate* μ ($\mu > 0$) the average number of cases that can be handled per time unit, and the *utilization* $\rho = \frac{\lambda}{\mu}$ is the expected fraction of time that the resource will be busy.

3 Modeling Workload Dependent Speeds in Terms of Colored Petri Nets

The effects of workload on processing speeds of resources were explored by simulating a CPN model. Colored Petri Nets (CPNs) is a high-level Petri net formalism that extends classical Petri nets with *data* (colored tokens), *time*, and *hierarchy* [8]. CPNs are bipartite directed graphs comprising of places and transitions and all the tokens in a particular place have a value of some common type. In CPN-terms, this means that all the tokens in a given place should belong to the same color set and each place has a color set (i.e., type). Tokens also have *timestamps* indicating when they can be consumed. When producing a token, it may be given a *delay*. This delay may be sampled from some probability distribution. Moreover, the time concept and the availability of many probability distributions in CPN Tools makes it possible to model performance aspects. By introducing resource tokens in the model, organizational aspects of the business process can be modeled. With the hierarchy concept of CPN Tools, it is possible to compose a CPN model in a modular way hence handling different levels of abstraction. CPNs can be distributed over so-called *pages* where one page can describe places and transitions and it may also refer to other pages [8]. Furthermore, CPN Tools have the ability to generate event logs which can be exported into ProM for analysis. Given the various functionality in CPN Tools necessary for modeling and simulating business processes, colored Petri nets were selected as the modeling language in this paper.

Our CPN model is a hierarchical model divided into six pages. The Overview (Fig. 4) page connects the Work Creation (Fig. 5) page and the Create Work Item page (Fig. 6). The CPN model handles tasks for a business process that

deals with the managing of travel requests at some university. We do not discuss this process in detail here, however, our model is generic to handle any other process model. The process model consists of a number of tasks and for each task we specify a number resources that are allowed to it. Cases are generated in the Work Creation page based on a predefined distribution and are put in the State List together with their initial state. The time between two subsequent case arrivals is given by the function IAT() and the creation time of cases is recorded by the current model time function Mtime(). After a case has been added to the Queue it is sent to the Create Work Item page.

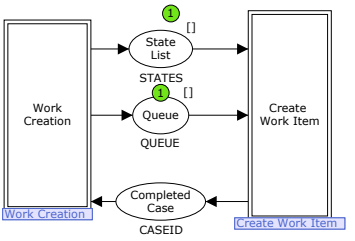

Fig. 4. The Main page

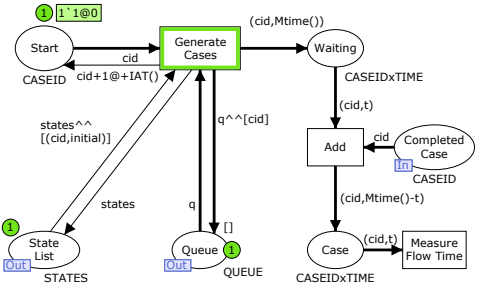

Fig. 5. The Work Creation page

Create Work Item. When the Queue arrives in the Create Work Item page in Fig. 6, Generate Random transition obtains the case that is at the top of the Queue and generates a random number used for making choices in the model where there are alternative paths have to be taken from a given place. The case together with the generated probability is added to the Case. Create Work Item transition takes a token from the Case and creates a new work-item. This transition obtains the next task to be executed for the case (using the State List and Process Declaration places), it also assigns a resource to execute the work-item (resources are obtained using predefined resource roles) and it also obtains the duration of the work-item. The work-item is added to the Work Item List which is sent to the Add Parameters page where resource parameters are calculated.

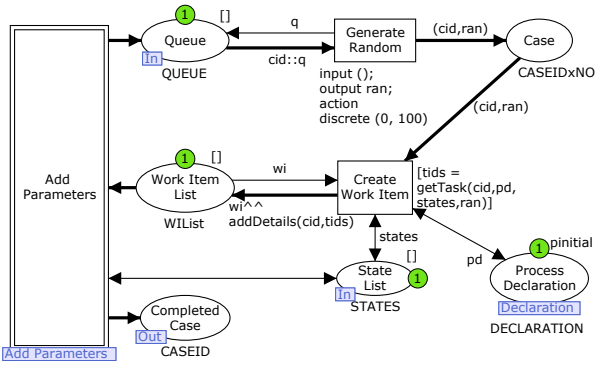

Fig. 6. The Create Work Item page

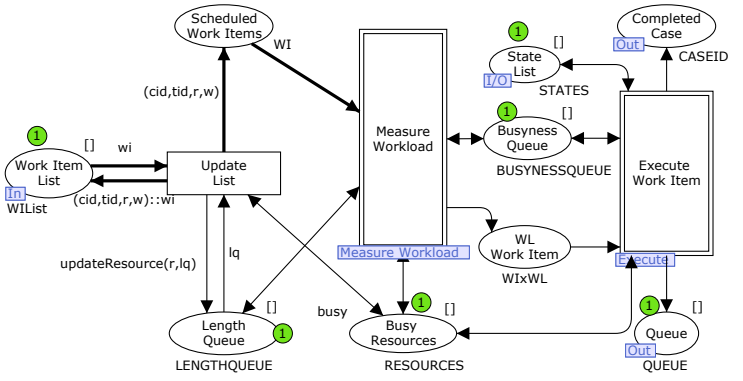

Fig. 7. The Add Parameters page

Add Parameters. When the Work Item List arrives in the Add Parameters shown in Fig. 7, Update List transition obtains each work-item from the Work Item List and adds it to the Scheduled WorkItems. Moreover, the Length Queue is also updated with information about the work-items that have been assigned to each resource. This is important because the information in the Length Queue will be used to determine the workload present in the model based on the *queue length* perspective. The Add Parameters page also connects the Measure Workload page (cf. Fig. 8) and the Execute Work Item page (cf. Fig. 9).

A work item is obtained from the Measure Workload page where its workload is measured and sent to the Execute Work Item page where it is actually executed. Moreover, if the work-item that has been completed in the Execute Work Item page is the last one for the case, then the case sent back to the Work Creation page through the Completed Case place.

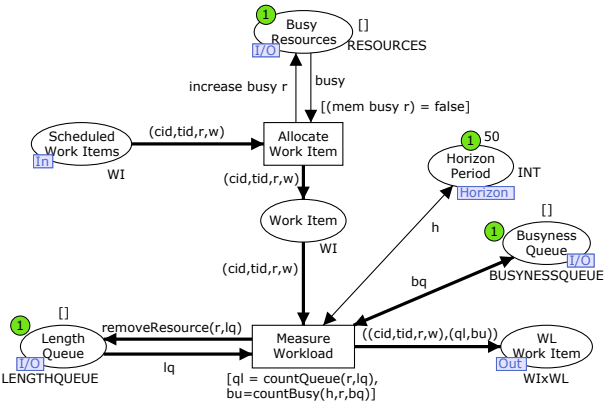

Fig. 8. The Measure Workload page

Measure Workload. In the Measure Workload page (Fig. 8), each work-item from the Scheduled Work Items is allocated to a resource to start its execution. This is only possible if the resource is not currently busy, i.e., is not in the Busy Resources. When Allocated Work Item transition fires, a work-item is taken from the list of Scheduled Work Items and added to the Work Item place. Moreover, the allocated resource will be added to the Busy Resources place. As soon as there is a token in the

Work Item place, Measure Workload will calculate the workload present before the allocated resource (r) starts execution. Measure Workload uses the information in the Length Queue and the Busyness Queues to calculate the workload present in the model.

- The Length Queue stores the number of work-items allocated to each resource. Therefore, function countQueue(r,lq) will return the number of work-items that have been scheduled for each resource (r) from the length queue (lq).
- For each resource, Busyness Queue will store a list for each that contains the the timestamps of work-items that each resource has completed so far. This information is used by the countBusy(h,r,bq) function to return the number of work-items from the busyness queue (bq) that have been completed in a specified period of time (h) for a resource (r). For example, if the given time period is $h = 100$, this function will returns the number of work-items completed during the previous 100 time units from the current time.

The measured workload information is added to work-item in the WL Work Item which is sent the Execute Work Item page. The number of work-items scheduled for the current resource are also reduced by 1 from the Length Queue.

Execute Work Item. The actual execution of a work-item is handled in the Execute Work Item page (Fig. 9). Here, the execution of the work-item is started and eventually completed. The duration of work-item processing is now dependent on the sampled duration from the Create Work Item page and the workload parameters from the Measure Workload page. We define a workload function $execTime(w, ql, bu)$ that takes the work-item duration (w), the queue length parameter (ql), and the how busy parameter (bu) to determine how long a resource will take executing a work-item. The function $execTime(w, ql, bu)$ uses the the power function to model the workload relationship and this implies with higher workload values the time of execution will be low. Moreover, the execution times will be higher if the amount of workload is low.

$$execTime(w, ql, bu) = w * 0.8^{((ql+bu)-1)} \tag{1}$$

Start Execution will move the work-item from the WL Work Item place to the Busy place. The length of work-item processing, i.e., done is given by the DurWL(w,ql,bu) function which is expressed in the CPN model but is equivalent to function shown in Equation 1. The current state for the case is updated in the State List based on the information in the pre and post sets from the process declaration. When Complete Execution transition fires, the resource is removed from Busy Resources and is now made available to execute scheduled work-items (if any). The State List is also updated with a new state for the case. The new state will have the number of tokens increased by 1 for the input place of task that will be executed next. If the task that has been completed is the last one for the case (given by finaltask), the case is sent back to the Work Creation page where its flow time is measured. Otherwise, it is added to the Queue and sent to the Measure Workload page where a new work-item for the case is created.

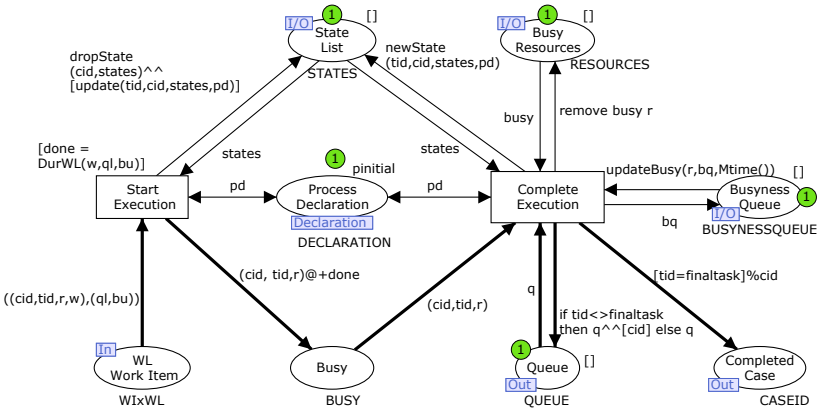

Fig. 9. The Execute Work Item page

The CPN model described creates an environment where we can analyze the impact of existing workload on the processing speeds of resources. In the next section we discuss experiments carried out to determine the effect of varying workload on the processing speeds of resources.

4 Simulation Experiments

Using the CPN model, experiments were carried out to investigate the effect of workload, i.e, denoted by the busyness and queue length perspectives on processing speeds, i.e., denoted by flow time. The monitor concept of CPN Tools allows measurement of performance indicators without changing the model [8] and were used to extract numerical data during the simulation experiments. The experimental results discussed here are based on simulations with 10 subruns, each subrun having 1,000 cases.

4.1 Experimental Results from Simulation Model

In this experiment, we compare the effect of varying arrival rates (λ) on the flow time values while keeping the service rates, i.e., $\mu = \frac{1}{15}$. The aim of this experiment is to compare the effect of increasing workload in the model on the flow time values and the utilization values. Considering the Yerkes-Dodson law shown in Fig. 2, we expect that the resource work slower when the workload is low but will eventually increase their speed as the workload increases in the model. Note that, although it is also possible for a busy resource to take longer while working, in our experiments we only focus on the notion that a busy resource works faster. This corresponds to the first half of the Yerkes-Dodson law.

We compare the flow time values obtained from two different input parameters that we use to determine the processing speeds of resources. In the first case, we consider experiments where we have assumed values for λ and μ. The

processing speeds of resources is entirely based on μ (sampled from an exponential distribution). In this case the value of done shown in Fig. 9 depends entirely on the sampled service rate, i.e., w. We refer to the results from this experiment as NoWL.

For the second case, the processing speeds of resources are dependent on the workload present in the system. The value for done (used to determine the duration of execution) is calculated from the $execTime(w, ql, bu)$ function defined in Equation 1. This function takes into account workload present based on the queue length and the how busy perspectives. We refer to the results from this experiment as WithWL.

Experiment 1: Effect of Varying Arrival Rates on the Flow Time. In the first experiment, we measure the effect of varying arrival rates on the average flow time. The results are shown in Fig. 10(a). In the case NoWL, as the arrival rates increase, the flow times also increase. Typically, the flow times dramatically increase when ρ get close to 1. The second case, WithWL, as the arrival rates increase the flow time values also increase. However, if we compare the curves for the experiments from NoWL and WithWL, we see that when workload is taken into account, initially the flow time values are higher than the situation without workload. This is because resources work slower since the workload in the system is low based on the lower arrival rates of cases. Moreover, as seen in Fig. 10(a) when the arrival rate values increase the flow time values also increase and eventually the resource works much faster and this leads to lower flow time values compared to the situation where workload is not taken into account (here the flow times are low at the start of the experiment and increase much higher as ρ tends to 1).

Experiment 2: Effect of Varying Arrival Rates on the Utilization. The second experiment shows the results on the average utilization values based on varying arrival rates. The experimental results are shown in Fig. 10(b). For

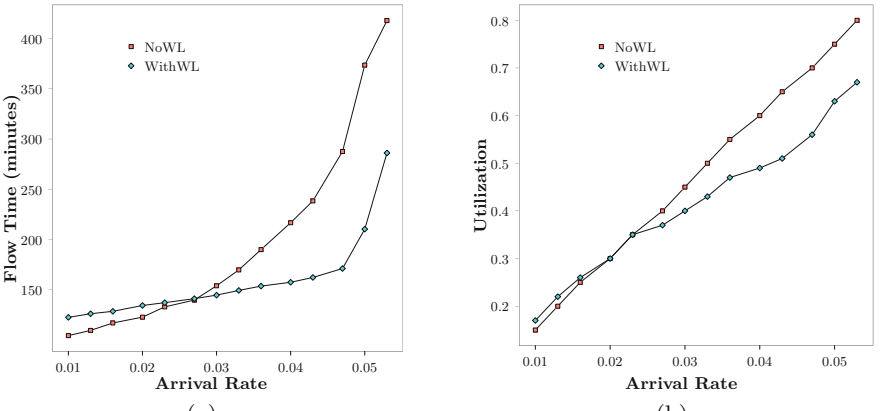

Fig. 10. The effect of varying arrival rates (λ) on (a) flow time and (b) utilization

comparison purposes, we obtained values for two scenarios NoWL and WithWL. Since the process model contains multiple resources, the values reported in this experiment are average utilization values. In the first case (NoWL), the utilization values follow a straight line because these are used as base line, for example, if $\mu = \frac{1}{15}$ and $\lambda = \frac{1}{18.75}$, then the expected utilization value is $\rho = 0.8$. However, if we compare the results from the second case with workload taken into account (WithWL), the utilization values are initially higher than the normal values because the resources work slower. However, as the arrival rates increase, yielding more workload in the system, the utilization values also increase but are lower than in the situation when resource work without workload effect on the processing speeds. This is because resources now work faster and hence will process more work-items. Therefore, the results from the experiments confirm that workload indeed has an effect on the processing speed of resources and eventually the utilization.

4.2 Analysis of Event Logs from a Real Life Process

This section is based on a real life case study taken from the CoSeLoG research project[1]. In the CoSeLoG project 10 Dutch municipalities are investigated as they execute their processes. The municipalities use workflow systems which generate event logs. The event log we use in this section is obtained from a process that deals with the handling of building permits. The log contains information about 324 cases, 19805 events, 15 resources and 175 activities. The start date of the log is "18-11-2009" and the end date is "12-01-2012". The results reported in this section are based on an extended implementation of a ProM plug-in described in [11] which quantifies the relationship between workload (using the queue length and how busy notions) and processing speeds (denoted by service and waiting times of events). Here, we use regression analysis to quantify the effect of workload (as the *independent variable* x),i.e., denoted based on queue length and how busy perspectives and processing speed (as the *dependent variable* y), i.e., denoted as service times of activities. The results are shown in Table 1 which include for each resource[2], the *correlation coefficient* (r) as the degree to which two variables are linearly related ($-1 \leq r \leq 1$) and the *r-square of the regression equation* (R^2, or the coefficient of determination), which is the proportion of variation in y accounted for by x [10].

The results in the table indeed confirm the effect of workload on the processing speeds of resources. Most of the R^2 shown in the table were higher than 0.7 which indicate that the variation in the service times is accounted for by the workload present in the system. For example, for resource "aweijter", "ckoets", "bwiling", their r and R^2 values were all high and this implies that as workload increases in the system, the speed of execution also increases. For, resources "cpers" in the 5th row of the table, their r and R^2 values were 0.97 and 0.94 respectively which implies a relationship between workload and the speed at which they worked.

[1] See http://www.win.tue.nl/coselog

[2] The resource names in Table 1 have been changed to ensure confidentiality.

Table 1. Linear regression results from real-life event log showing the relationship between workload and execution times of resources

resource	correlation coefficient	R^2	resource	correlation coefficient	R^2
aweijter	0.88	0.78	bwiling	1.00	1.00
ckoets	0.97	0.94	cokti	0.78	0.61
ejansen	0.92	0.85	spalm	0.9	0.81
phendric	1.00	1.00	brows	0.99	0.98
tjansen	0.99	0.98	cpers	0.76	0.58
mpauel	0.73	0.53	gursel	0.77	0.6

The results discussed in this section follow the first half of the Yerkes-Dodson law. Given that the simulation model generates event logs, it is now possible to analyse those logs using the process mining techniques that we have described in this section and in our earlier work [11].

Although varying workload has an effect on the speed at which resources work, when building simulation models this is rarely taken into account. From the experimental results discussed in this section, we see that workload has an impact on the flow times of cases, so if we assume that workload has no effect on the processing speeds, we may get performance results not corresponding to reality. Moreover, it is interesting that we not only show how to model such resource aspects in simulation models, but that such phenomena also exists in real life. Since, the event logs contains precise timestamps, we have been able to empirically investigate and quantify the effect of workload on processing speeds. Therefore, resources do not always work at constant speeds and there can be a number of factors that influence their speeds and in this section we shown that workload based queue length and the how busy perspectives indeed influences the resource processing speeds.

5 Related Work

The work presented in this paper is related to earlier work that has been done in the field of operations management and in simulation studies. The "Yerkes-Dodson Law of Arousal" [16] illustrated in Fig. 2, is one of the main motivations for this paper. Although the Yerkes-Dodson law originally related arousal to performance, this law has been extended to incorporate workload in the place of arousal. This law nominally depicts a low performance when the resource is over-worked or under-worked. In operations management, substantial work has been done to operationalize this "law" using mathematical models. Earlier work supporting this law shows that there exists a relationship between work-in-process and productivity [5,7]. The authors in [7] shows that the basis of industrial statistics collected that a strong correlation exists between productivity improvement and the speed of industry networks. In [13] queues with workload-dependent

arrival and service rates are considered and the authors characterize the effect of work-in-process (the number of work orders on the shop floor) on productivity (the output per employee).

The discovery of simulation models is an approach that has been presented in [15]. Given an event log, a simulation model can be discovered and simulated using ProM and CPN Tools. In [9] a similar approach is presented that shows how to redesign a business process and predict its future performance based on simulation. The resource models used in both approaches are very simple, for example, the authors in [9] do not say anything about resources and their involvement in the simulation and redesign of business processes. However, to truly map a business process onto a simulation model, it is crucial that the resource perspective is modeled accurately. Although we focus on workload speeds in this paper, our earlier work also takes into account that people are involved in multiple processes, are available only part-time, and work in batches. Experiments show that these factors really influence performance [3].

The work in this paper is also related to earlier work presented in [11] where process mining techniques are used to analyse the relationship between workload and processing speeds based on event logs. In this paper, we focus on log generation from simulation models with workload dependent speeds.

6 Conclusion

Although organizations use simulation to analyze their business processes, the results may be very misleading if the assumptions used are incorrect. In this paper, we have addressed one of the problems discussed, i.e., human resources are often modeled incorrectly in simulation models. It is shown that resources are typically modeled in a naive manner and that this highly influences the simulation results. We have focussed in improving simulation models by providing a better modeling of resource behavior. The fact that people do not work at constant speeds and that workload has an effect on the processing speeds of resources has effects on the key performance indicators of a process. We characterized workload based on the queue length and how busy perspectives.

Based on a CPN model and with experiments, we show that the workload present in a system affects the processing speeds of resources. The information about the behavior of resources and also results from our earlier work [3, 11] show that these characteristics should not be ignored when building simulation models. Moreover, simulation models can generate event logs contain information characterizing the workload relationship. This relationship is hidden, but correlations between workload and processing speeds can be observed using process mining techniques [11]. It is important to use both simulation and process mining techniques to better understand, model, and improve real-life business processes. Moreover, the relation can also be used as a basis for providing operational support where users are provided with recommendations about the next actions to take [4, 12].

References

1. van der Aalst, W.M.P.: Business Process Simulation Revisited. In: Barjis, J. (ed.) EOMAS 2010. LNBIP, vol. 63, pp. 1–14. Springer, Heidelberg (2010)
2. van der Aalst, W.M.P.: Process Mining-Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)
3. van der Aalst, W.M.P., Nakatumba, J., Rozinat, A., Russell, N.: Business Process Simulation. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1: Introduction, Methods and Information Systems. International Handbooks on Information Systems, pp. 313–338. Springer (2010)
4. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond Process Mining: From the Past to Present and Future. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 38–52. Springer, Heidelberg (2010)
5. Bertrand, J.W.M., van Ooijen, H.P.G.: Workload Based Order Release and Productivity: A Missing Link. Production Planning and Control 13(7), 665–678 (2002)
6. CPN Tools, cpntools.org (2012)
7. Holström, J.: The relationship between speed and productivity in industrial networks: A study of industrial statistics. International Journal of Production Economics 34, 91–97 (1994)
8. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modeling and Validation of Concurrent Systems. Springer, Berlin (2009)
9. Maruster, L., van Beest, R.T.P.: Redesigning Business Processes: A Methodology Based on Simulation and Process Mining Techniques. Knowledge Information Systems 21, 267–297 (2009)
10. Montgomery, D.C., Peck, E.A.: Introduction to Linear Regression Analysis. Wiley & Sons (1992)
11. Nakatumba, J., van der Aalst, W.M.P.: Analyzing Resource Behavior Using Process Mining. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 69–80. Springer, Heidelberg (2010)
12. Nakatumba, J., Westergaard, M., van der Aalst, W.M.P.: An Infrastructure for Cost-Effective Testing of Operational Support Algorithms Based on Colored Petri Nets. In: Proceedings of the 33rd International Conference on Applications and Theory of Petri Nets. LNCS. Springer (2012)
13. van Ooijen, H.P.G., Bertrand, J.W.M.: The Effects of a Simple Arrival Rate Control Policy on Throughput and Work-in-progress in Production Systems with Workload Dependent Processing Rates. International Journal of Production Economics 85, 61–68 (2003)
14. ProM (2012), `http://www.processmining.org`
15. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering Simulation Models. Information Systems 34(3), 305–327 (2009)
16. Wickens, C.D.: Engineering Psychology and Human Performance. Harper (1992)