Advanced Simulation of Resource Constructs in Business Process Models

Sander P. F. Peters(✉), Remco M. Dijkman, and Paul W. P. J. Grefen

Eindhoven University of Technology, Eindhoven, The Netherlands
{s.p.f.peters,r.m.dijkman,p.w.p.j.grefen}@tue.nl

Abstract. Simulation is often used as a tool to assess the performance of business processes. However, current business process simulation engines do not support advanced resource constructs, such as work allocation strategies and case attributes. Using only basic resource constructs leads to performance metrics that deviate significantly from the real process performance. Therefore, a clear need arises for simulation engines that incorporates advanced resource constructs. Addressing this need, we present the resource patterns that should be supported by simulation engines, a conceptual model to support them, and a prototype implementation of this conceptual model. The model and engine are evaluated in a simulation experiment that highlights utilization rates under different conditions. This experiment shows that the advanced resource constructs significantly outperform the basic resource constructs. From this we can also conclude that existing simulation engines must be extended with advanced resource constructs to properly simulate processes from practice that use these constructs.

1 Introduction

Analysis of the performance and feasibility of business processes is important to evaluate the effects of business process reengineering and redesign efforts [6]. In order to assess the performance of these business process models, simulation is often used. However, Recker [15] shows that more research into the simulation of business process models is needed.

First, business process simulation engines should take into account basic simulation parameters, as described in the standard work on simulation by Law and Kelton [12]: each simulation engine should support a warm-up period, replications and confidence intervals. A warm-up period ensures correct simulation results, since the empty system at the beginning of the simulation can pollute the final performance of the model. In order to obtain confidence intervals using the central limit theorem, replications need to be in place in the simulation engine.

Second, a strong resource perspective is needed to ensure that the process models correctly represent reality. However, we will show in Sect. 2 that commonly used business process simulation engines have problems coping with advanced resource constructs, such as resource dependencies and queueing strategies. We focus on simulation engines that use the Business Process Model

M. Weske et al. (Eds.): BPM Forum 2018, LNBIP 329, pp. 159–175, 2018.
https://doi.org/10.1007/978-3-319-98651-7_10

and Notation (BPMN) [14], because this has become the de-facto standard for modeling business processes. BPMN mainly focuses on the control-flow perspective of a process, while the resource perspective in the BPMN language is limited to the lanes and pools concepts [19]. In contrast, business process execution languages like YAWL [20] have extensive support for the resource perspective, which indicates that there is a need for supporting the resource perspective. The advanced resource patterns used in these execution languages are also defined in Russell et al. [18].

The consequence of the shortfalls of BPMN simulation engines, is that the results obtained by them will be less valid for the real-life process that is simulated. In our evaluation we will indeed show that BPMN models, in which advanced resource patterns are not included, have simulation results that deviate significantly from models, in which they *are* included. Consequently, we can conclude that BPMN models cannot be used to draw valid conclusions for business processes from practice that use advanced resource constructs, such as the case handling principle.

Therefore, the aim and contribution of this paper is to present the conceptual model and behavior of a simulation engine that supports the advanced simulation requirements outlined above. We also present a prototype implementation of this engine. This contributes to laying the foundation for the simulation of complex business processes with the complete inclusion of resources for real world processes, which can enhance business process engineering and redesign efforts.

Against this background, the remainder of this paper is structured as follows. Section 2 describes the state of the art of existing business process simulation engines. In Sect. 3 an extension of process models is proposed to make them fit for simulation purposes, solving the limitations of the existing models. In Sect. 4 the formal behavior of the new simulation engine is provided by means of a state-space and state-space transitions. Section 5 describes the implementation of the simulation engine from Sect. 4 and the evaluation of the engine. The focus in this section is on the effect of resource dependencies on the utilization of resources. In Sect. 6 the conclusion based upon the evaluation of the new simulation model is presented and future work is discussed.

2 Related Work

Table 1 provides the overview of the support of existing simulation engines for advanced simulation concepts. In order to assess the selected simulation engines, several criteria are taken from literature. These criteria focus on the support a simulation engine provides for the simulation of business processes with a special focus on the resource perspective. For each feature available in a tool there is a + symbol. If a feature is not available in a tool a − symbol is used. For cases where a feature can be found, but only in the form of a complex workaround or in a limited manner, there is a ± symbol.

First basic flow criteria are assessed, based upon the paper of Van der Aalst et al. [24], three basic criteria are selected: sequence of activities, parallel execution of activities and branching or choice in activities. These have been chosen

Table 1. Analysis of simulation engines

	Tool:	*Arena*	*CPN*	*Adonis*	*BIMP*	*Bizagi*	*BPSim*	*IBM*	*Signavio*	*TIBCO*	*Prototype*
Basic Flow	Sequence	+	+	+	+	+	+	+	+	+	+
	Parallelism	+	+	+	+	+	+	+	+	+	+
	Branching	+	+	+	+	+	+	+	+	+	+
Activities	Arrival Distributions	+	+	+/−	+	+	−	+	−	+	+
	Duration Distributions	+	+	−	+	+	−	+	+/−	+	+
	Resource Requirements	+	+	+	+	+	+	+	+	+	+
	Cost per Activity	+	+	+	+	+	+	+	+	+	+
Resources	Capacity	+	+	+	+	+	+	+	+	+	+
	Roles	+	+	+	+	+	+	+	+	+	+
	Schedules	+	+	+	+	+	+	+	+	+	+
	Cost of Usage	+	+	+	+	+	+	+	+	+	+
	Multiple Roles	−	+/−	+	−	−	−	+	−	+	+
Advanced Con-structs	Queueing Strategies	+	+/−	−	−	−	−	−	−	−	+
	Separation of Duties	+/−	+/−	−	−	−	−	−	−	+	+
	Case Handling	+/−	+/−	−	−	−	−	−	−	−	+
	Allocation Strategies	−	+/−	−	−	−	−	−	−	−	+
	Case Attributes	+	+	−	−	−	−	−	−	+	+
Simulation	Duration	+	+	+	+	+	+	+	+	+	+
	Warm-Up Period	+	+	−	−	−	−	+	−	−	+
	Replications	+	+	+	−	+	−	−	−	−	+
	Confidence Intervals	+	+	−	−	−	−	−	−	−	+

since these are in the top 10 of most used constructs [29]. Next to these basic flow criteria, the paper of Tumay [22] indicates that also activities and resources are basic elements in a simulation. Tumay [22] shows that each simulation needs an arrival process, also each activity can be associated with a duration and a cost function. The duration of an activity and arrival rate can consist of mathematical distributions. Activities can also require resources to be executed in the process model. Resources, according to the paper of Russell et al. [18], should be modeled having a certain capacity, role and schedule. Also resources may have costs of usage associated with them. Therefore all simulation engines are assessed on these criteria for activities and resources.

The paper of Russell et al. [18] also defines several different allocation strategies for resources, in this survey the focus will be on the constructs of separation of duties and case handling from the patterns defined. According to Russell et al. [18] separation of duties is described by as: "*The ability to specify that two tasks must be allocated to different resources in a given workflow case*" and case handling is described as: "*The ability to allocate the work items within a given workflow case to the same resource*". The paper of Tumay [22] indicates that queueing strategies are important for the way entities are sequenced in queues at activities. Furthermore allocation strategies and case attributes are analyzed as important aspects in the simulation [18]. Next to these specific business process simulation properties, general properties of simulation engines will be reviewed in terms of duration of the simulation, ability to incorporate warm-up time, use of replications and the reporting of statistics making use of confidence intervals. These properties are prescribed by Van der Aalst et al. [23] as the main important properties to create a sound simulation experiment.

The simulation engines which are selected for this survey are composed of two types of tools: general purpose simulation tools and business process simulation tools, with the main focus on BPMN simulation engines. Based upon the paper from Jansen-Vullers and Netjes [7] two general purpose simulation tools are identified as Arena [9] and CPN Tools [8]. Furthermore all tools mentioned in the book Fundamentals of BPM [5] in Chap. 7.4.3 Simulation Tools will be assessed. Unfortunately ARIS, OpenText, Oracle and Savvion have no public available simulation tool to assess and therefore are excluded from the survey. Also the developed prototype as discussed in this paper will be scored on the criteria presented in Table 1.

As can be seen in Table 1 the two general purpose simulation tools perform quite well on mainly all aspects, only on the resource allocation rules they score a ±. In contrast the regular business process simulation tools perform worse on the resource allocation rules, where barely any advanced construct is supported. Furthermore there are major issues in general simulation properties, lack of confidence intervals, replications and warm-up periods makes most tools unfit for proper simulation according to the paper of Van der Aalst et al. [23]. General purpose simulation tools are less in favor than the business process simulation tools, which are easier to use and do not take a steep learning curve to model processes, since they use the BPMN modeling language [21,27].

3 Advanced Simulation Model

The basic idea to simulating a business process model is to transform it into a queueing network [12]. In order to create that network, each activity in the BPMN model is transformed into a queue that contains cases on which that activity must be performed. Whenever the activity has been performed for a case in the queue, the case is put into the queue of the next activity that should be performed on it, conform the behavior that is described by the BPMN model.

In addition to this basic translation, which is common for all BPMN simulation engines, we propose constructs to simulate the following advanced constructs: case attributes, resource schedules, resources acting in multiple roles, queueing principles, resource dependencies, and simulation parameters. For illustration purposes, Fig. 1 shows a BPMN model that includes these advanced constructs. We explain these constructs in the following paragraphs.

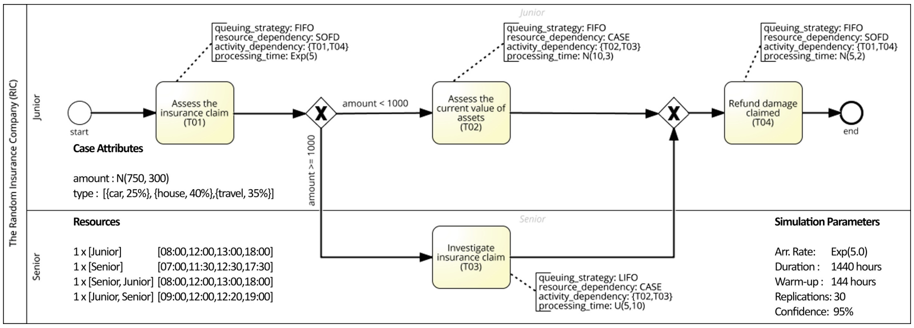
The Random Insurance Company (RIC)
Junior
Senior
queuing_strategy: FIFO
resource_dependency: SOFD
activity_dependency: {T01,T04}
processing_time: Exp(5)
queuing_strategy: FIFO
resource_dependency: CASE
activity_dependency: {T02,T03}
processing_time: N(10,3)
queuing_strategy: FIFO
resource_dependency: SOFD
activity_dependency: {T01,T04}
processing_time: N(5,2)
start
Assess the insurance claim (T01)
amount < 1000
amount >= 1000
Assess the current value of assets (T02)
Refund damage claimed (T04)
end
Case Attributes
amount : N(750, 300)
type : [{car, 25%}, {house, 40%},{travel, 35%}]
Resources
1 x [Junior] [08:00,12:00,13:00,18:00]
1 x [Senior] [07:00,11:30,12:30,17:30]
1 x [Senior, Junior] [08:00,12:00,13:00,18:00]
1 x [Junior, Senior] [09:00,12:00,12:20,19:00]
Investigate insurance claim (T03)
queuing_strategy: LIFO
resource_dependency: CASE
activity_dependency: {T02,T03}
processing_time: U(5,10)
Simulation Parameters
Arr. Rate: Exp(5.0)
Duration : 1440 hours
Warm-up : 144 hours
Replications: 30
Confidence: 95%

Fig. 1. BPMN Model with advanced parameters

3.1 Case Attributes

A case is an executable instance of a process, where each case can have certain attributes, also known as case data [28]. In practice, the choice between alternative paths through a process depends on the values of the case data. However, current simulation engines solve a choice between alternative paths by placing probabilities on the paths themselves. The benefit of associating alternative paths with the values of case data, is that this allows for interaction between multiple alternative paths to be captured.

To facilitate this, we define case attributes on the process level, where each attribute has a name and a value associated with it. Values for different attributes can be obtained from the probability distributions of the attributes as specified in the process model. Distributions can be either numerical or categorical, numerical distributions can be the normal distribution, the exponential distribution, the uniform distribution or a static value in this simulator. Categorical variables are uniformly distributed variables, where the attribute is assigned one of the values of the categorical variable. All values for the case are determined at the arrival of the case at the process. For example a case of an insurance claim process is of the category car (0.25), house (0.40) or travel (0.35) and has a certain amount of money claimed, which is normally distributed with an average of 1750 and a variance of 500.

3.2 Resource Schedules

According to Russell et al. [18] resources need to have a schedule. A schedule is an overview of the periods during which a resource is available for processing cases, also called the active state. When a resource is outside of the scheduled times, it is passive. During the time a resource is passive it will not actively participate in the execution of the business process. Each schedule is denoted by a chain of times in a day when a resource changes its state. For the first time in the schedule the resource becomes active and at each following event the state will change to passive if the resource was active and the state will change to active when the resource was passive. We will represent a schedule as a list of times at which the resource changes state from active to passive or vice versa, assuming that the resource starts passive, for example [08:00, 12:00, 13:00, 18:00]. Each resource will finish the last activity, if necessary in overtime, before becoming passive according to the schedule.

3.3 Resources Acting in Multiple Roles

According to Van der Aalst, Weske and Grünbauer [25] a resource can have multiple roles, based on the organizational model of the enterprise. To facilitate this, we define a resource as having a certain resource type, which is a list of roles belonging to a resource. A resource type can be a single role or a combination of multiple roles which exist in the same pool in a process model. Furthermore each of the resource types has an integer which represents the number of instances of the resource type. Each instance of a resource type is able to serve the roles which are associated with the type in the process and will stick to the order of the roles for prioritizing roles. For example if there are two roles, say A and B, resources can be of type [A], type [B], type [A,B] or type [B,A]. The quantities of resource types are defined as part of the process.

3.4 Queueing Principles

Queueing principles are the order in which a resource selects a case from a queue to handle it, this is also called the service discipline of the queue. The queue is the worklist which contains the work items offered to the resource. According to Righter et al. [16] queues can have different service disciplines, such as First In First Out (FIFO) and Last In First Out (LIFO). In the FIFO system the case first in the queue is also the first one to be handled by the resource. Where with the LIFO system the last case in the queue is handled first by the resource. Next to those two variants a random selection can be applied as a service discipline, in which the next case to be handled is selected randomly, or a priority system can be used to sort the queue according to the priority value of each case [12].

3.5 Resource Dependencies

Resource dependencies are specific dependencies on resource instances between activities in a process. From the paper of Russell et al. [18] two resource dependencies between activities can be identified: separation of duties (SOFD) and case handling (CASE). A resource dependency is an extra constraint on which resources may handle the activity. The separation of duties dependency yields that a different resource instances should execute the activities affected by the dependency. For the case handling dependency it is the other way around, it yields that the same resource instance should execute the activities affected by the dependency. Regularly each activity is not affected by any resource dependency and can be executed on its own by a resource instance. When there are resource dependencies between multiple activities for each activity in the dependency it should be denoted which type of dependency applies and which other activities are involved. For example if the activities A02 and A04 are subject to the resource dependency case handling both will have the resource dependency label CASE and the associated set of activities {A02, A04} in their properties denoted. If no resource dependency label is provided it is assumed that there is no resource dependency in the model for the given activity.

3.6 Simulation Parameters

In order to perform simulation, several main components should be defined in the BPMN model. According to Van der Aalst et al. [23] and Law and Kelton [12] the duration of the simulation, warm-up period, number of replications and confidence intervals are vital components of a simulation model. The warm-up period of the simulation is the amount of time where no performance metrics are recorded at the start of a simulation run [13]. This is needed because steady state simulation models start empty and idle, this empty state influences the performance metrics and therefore need to be corrected for. Replications are the repetition of a simulation with fixed inputs but different outputs due to different random number streams [3], these are needed to obtain reliable values for performance metrics. Using the values from the different replications confidence intervals can be computed using the central limit theorem [23]. These confidence intervals are a limited bandwidth where the actual value of a performance metric can be approached with a certain probability. These parameters should be set at the model level of a BPMN model in order to enable proper execution.

4 Advanced Simulation Behavior

In this section we formally define the behavior of a discrete event simulation that can simulate the concepts that are defined in the previous section. We define the behavior in terms of the statespace of a simulation model as it is specified in the previous section, the initial state of that model and the state transitions that the simulation model will take from the initial state.

The behavior is defined for the meta-model that is implicitly defined in the previous section and exemplified in Fig. 1. Due to space limitations, we only postulate the existence of the concepts that we need from the meta-model.

Definition 1 (Model Elements). *Let $\mathcal{I}$ be the set of all possible identifiers and $id : \mathcal{I}$ the function that returns a new unique identifier. Furthermore, let $\mathcal{L}$ be the set of all possible labels, $\mathcal{V}$ be the set of all possible data values, $\mathcal{S}$ be the set of all possible schedules and $\mathcal{D}$ be the set of all possible probability distributions.*

A simulation model defines a set of activities $\subseteq \mathcal{L}$ *and a set of* attributes $\subseteq \mathcal{L}$ *by their labels. It describes a set of* resourcetypes $\subseteq \mathcal{L} \times \mathbb{P}(\text{activities}) \times \mathcal{S} \times \mathbb{N}$ *(where $\mathbb{P}$ represents the powerset), such that each resource type $(l, as, s, n) \in$* resourcetypes *has a label l, a set of activities as that a resource of that type* *can perform, a schedule s for the availability for resources of that type, and a number of resources n of that resource type. A schedule is a list of timestamps at which a resource changes state, a function to retrieve the next time of a state change is $t = getnext(s, currenttime)$, which returns the next time in a schedule at which the resource changes state. In addition a simulation model describes: a processing time distribution for each activity a, denoted as* dist_a*, an interarrival time distribution, denoted as* dist_i*, and a distribution for the values of each case attribute d, denoted as* dist_d*. We define the existence of a function* $\text{sample} : \mathcal{D} \rightarrow \mathcal{V}$ *that samples a value from a distribution.*

Furthermore, a BPMN model describes a behavior that can be defined in terms of a token-game, as explained for example in Van Gorp and Dijkman [26] and Dijkman, Dumas and Ouyang [4]. Here, we assume the existence of functions that implement the token game. These functions should be implemented conform the BPMN semantics, as defined in these papers.

Definition 2 (Model Behavior). *Let $\mathcal{M}$ be the set of all possible markings in a token game on a BPMN model. Given a particular model, there is: a function* $\text{initialmarking} : \mathcal{M}$ *that returns the initial marking of the model for a starting case, a function* $\text{enabled} : \mathcal{M} \rightarrow \mathbb{P}(\text{activities})$ *that returns the set of activities that can happen in a given marking, and a function* $\text{fire} : \mathcal{M} \times \text{activities} \rightarrow \mathcal{M}$ *that defines the marking to which a model transitions when a particular activity happens in a particular marking.*

With these definitions, we can define the statespace of the simulation model. Since a discrete event simulation is based on a queueing network, we define an abstract 'queue' type.

Definition 3 (Queue). *Let $queue$ and $queue'$ be two queues and e be a potential element of a queue, we postulate a the existence of functions and constants: $[]$, which represents the empty queue; $queue' = put(queue, e)$, which places the element e in the queue, resulting in a new queue (the exact position at which the element e is placed in the queue may depend on the queueing policy, e.g. in a FIFO queue the element e is placed at the end, but in a LIFO queue the element e is placed at the start); $e \in queue$, which returns true is the element e is in the queue; $e = first(queue)$, which returns the first element in the queue; $queue' = removefirst(queue)$, which removes the first element from the queue; and $queue' = remove(e, queue)$, which removes element e from the queue.*

The parts of the statespace are: the queues that contain the cases that are queueing for a certain activity to be performed on it; the queues that contain the resources that are waiting to perform an activity that they can perform according to their resource type; a queue of simulation events that still have to occur in the simulation; and the current simulation time. Consequently, we define these elements as follows.

Definition 4 (Case, Resource, Event, Statespace). *Let $\mathcal{T}$ be the set of possible timestamps.*

A ***case*** *is a tuple (id, as), in which: $id \in \mathcal{I}$ is a unique identifier; $as \subseteq attributes \times \mathcal{V}$ is a set that assigns values to attribute labels for the case.*

A ***resource*** *is a tuple (id, rt), in which: $id \in \mathcal{I}$ is a unique identifier; and $rt \subseteq \mathcal{L}$ is the label of the resource type of this resource.*

A simulation ***event*** *is a tuple $(t, type, i)$, in which: $t \in \mathcal{T}$ is the timestamp at which the event is scheduled to occur; $type \in \{\text{Arrival}, \text{Complete}, \text{Activate}, \text{Passivate}\}$ is the type of simulation event, which can either be the arrival of a new case in the system, the completion of an activity for a case, an activation of a resource or pacification of a resource; and i is the information that is associated with the event, which is a case if $type = \text{Arrival}$, a resource if $type = \text{Activate}$ or Passivate, a tuple (r, c, a) if $type = \text{Complete}$, in which a is the activity that was just completed, r is the resource that was performing the activity and c is the case for which the activity was being performed.*

By convention, we refer to elements of a tuple by the labels that are defined for them above. E.g. for an event e, e_{type} refers to the type of e.

We can now define the statespace of a simulation model as follows.

Definition 5 (Statespace). *The statespace is of a simulation model is composed of:*

- *$foreach$ activity a, a queue $queue_a$ of cases that are queueing for that activity;*
- *$foreach$ resourcetype rt, a queue $queue_{rt}$ of resources that are queueing to perform activities;*
- *a queue $queue_s$ of simulation events ordered by their timestamp with the event with the lowest timestamp first in the queue;*
- *a function $states : case \rightarrow \mathcal{M}$ that associates cases with their current marking in the BPMN model;*
- *a set of $assignments \subseteq resource \times case$ of resources currently working on a case;*
- *the $currenttime$ of the simulation model; and*
- *a set $passiveresources$ of resources that are passive.*

An example of a statespace is provided in Table 2. Six cases and five resources are active in the model. For activities where no resource is available, the cases are queued in the activity queues. For resources where all associated activity queues are empty are queued in the resource queues. The marking is not specifically defined, we illustrate the marking as activities marked with cases.

We can now define the algorithms that set the initial state of the simulation and that describe the behavior of the simulation. The initial state of the simulation is constructed through Algorithm 1. The algorithm creates: an empty queue of waiting cases for each activity; a queue of resources for each resource type, containing as many resources as specified by the resource type; the queue of simulation events, containing one new case arrival event and activation events

Table 2. Example of a statespace

Object	Value
Cases	c_1 = (13,{(amount,1293),(type,'car')}), c_2 = (14,{(amount,144),(type,'travel')}) c_3 = (15,{(amount,1325),(type,'house')}), c_4 = (16,{(amount,749),(type,'car')}) c_5 = (17,{(amount,635),(type,'house')}), c_6 = (18,{(amount,125),(type,'travel')})
Resources	r_1 = (1, [junior]), r_2 = (2,[senior]), r_3 = (3,[senior, junior]), r_4 = (4,[junior,senior]), r_5 = (5,[senior])
Activity queues	$queue_{T01}$: [c_5,c_6]; $queue_{T02}$: [ ]; $queue_{T03}$: [ ]; $queue_{T04}$:[c_1]
Resource queues	$queue_{jr}$: [ ]; $queue_{sr}$: [r_2]; $queue_{sr,jr}$: [ ]; $queue_{jr,sr}$: [ ]
$queue_s$	[(00:12:45, Arrival, (19,{(amount,855),(type,'travel')})), (00:14:22, Complete, (r_1,c_4,T02)), (00:15:37, Complete, (r_3, c_2,T04)), (00:16:58, Complete, (r_4,c_3,T03)), (00:17:00, Passivate, r_1), (00:17:15, Passivate, r_2) (00:17:45, Passivate, r_3), (00:18:00, Passivate, r_4), (00:36:00, Activate, r_5)]
Marking	{(T04,c_1), (T04,c_2), (T03,c_3), (T02,c_4), (T01,c_5), (T01,c_6)}
Assignment	{(r_1,c_4), (r_3,c_2), (r_4,c_3)}
currenttime	00:11:39
passiveresources	{r_5}

Algorithm 1. Create the initial state of the simulation model

```
1: **for all** activity $a \in$ activities **do** $queue_a = []$
2: $newcase = (id(), \{(a, sample(dist_a))|a \in attributes\})$
3: $states(newcase) = initialmarking$
4: $currenttime = 0$
5: $queue_s = put([], (sample(dist_i), Arrival, newcase))$
6: **for all** resourcetype $(rt, as, s, n) \in$ resourcetypes **do**
7:     $queue_s = put(queue_s, (getnext(s, currenttime), Activate, rt))$
8: $assignments = \emptyset$
9: $passiveresources = \emptyset$
10: **for all** resourcetype $(rt, as, s, n) \in$ resourcetypes **do**
11:     $queue_{rt} = []$
12:     **for** i = 1 to n **do** $passiveresources = passiveresources \cup \{(id(), rt)\}$
```

for all resources; an empty set of assignments; and a set of passive resources, which initially contains all resources.

After the initial state, the simulation events in the event queue are processed by the simulation engine. Algorithm 2 specifies how this works. The algorithm processes simulation events in the order in which they should occur. If the sim-

Algorithm 2. Simulating the model by processing simulation events

```
 1: while currenttime < endtime do
 2:     $(t, type, i) = first(queue_s)$
 3:     $queue_s = removefirst(queue_s)$
 4:     $currenttime = t$
 5:     if type = Arrival then
 6:         for all $a \in enabled(states(i))$ do $queue_a = put(queue_a, i)$
 7:         $newcase = (id(), \{(a, sample(dist_a)) | a \in attributes\})$
 8:         $states(newcase) = initialmarking$
 9:         $queue_s = put(queue_s, (currenttime + sample(dist_i), Arrival, newcase))$
10:     else if type = Complete then
11:         $assignments = assignments - \{(i_r, i_c)\}$
12:         $queue_{rt'} = put(queue_{rt'}, i_r)$, where $rt' = i_{r_{rt}}$
13:         $states(i_c) = fire(states(i_c), i_a)$
14:         for all $a \in enabled(states(i_c))$ do $queue_a = put(queue_a, i)$
15:     else if type = Activate then
16:         $passiveresources = passiveresources - \{i\}$
17:         $queue_{rt'} = put(queue_{rt'}, i)$, where $rt' = i_{rt}$
18:         $queue_s = put(queue_s, (getnext(rt'_s, currenttime), Passivate, i)$
19:     else if type = Passivate then
20:         if $(r, c) \in assignments$, such that $i = r$ then
21:             find $(t, Complete, (r', c', a)) \in queue_s$, for which $r' = r$ and $c' = c$
22:             $queue_s = put(queue_s, (t, Passivate, r))$
23:         else
24:             $queue_{rt'} = remove(queue_{rt'}, i)$, where $rt' = i_{rt}$
25:             $passiveresources = passiveresources \cup \{i\}$
26:             $queue_s = put(queue_s, (getnext(i_{rt_s}, currenttime), Activate, i))$
27:     for all $a \in activities$, for which there exists $c \in queue_a$ and a resource type
        $(rt, as, n) \in resourcetypes$, such that $queue_{rt} \neq []$ and $a \in as$, using a fair distri-
        bution of resources over activities do
28:         $r = first(queue_{rt})$
29:         $queue_a = removefirst(queue_a)$
30:         $assignments = assignments \cup \{(r, c)\}$
31:         $e = (currenttime + sample(dist_i), Complete, (r, c, a))$
32:         $queue_s = put(queue_s, e)$
```

ulation event is an arrival event, the case that arrives is put in the queues of the activities that are enabled in the current (initial) marking of the case. Subsequently, an event is generated for the next arrival. If the simulation event is a completion event, the resource processing the completed activity is released, the next marking of the case is computed, and the case is put in the queues of the activities that are enabled in that marking. If the simulation event is an activate event, the passive resource is activated and a new passivate event is scheduled. If the simulation event is a passivate event, first there is checked whether the resource which should become passive is currently processing an activity for a case, if so a new passivate event is scheduled at the completion time of the activity. If the resource is not currently processing an activity, it is removed from the resource queue and moved to the passiveresources and a new activate event is scheduled. Subsequently, it is determined if there are any activities on waiting cases that can now be performed, because there are resources available to do so. Activities must be processed in an order that is 'fair', for example longest-waiting-time-first or round-robin.

5 Evaluation

The advanced simulation engine that is defined in this manner, improves on current BPMN simulation tools. Consequently, the hypothesis of this paper is that the current BPMN simulation tools produce analysis results that significantly deviate from reality.

In order to test this hypothesis, a simulation experiment will be conducted. In this paper the focus will be on resource dependencies, which consist of the concepts of separation of duties and case handling. In future work we will also investigate the effect of different queueing principles, case data, and other advanced resource patterns.

5.1 Simulation Methodology

The evaluation of the effect of the different resource dependencies is performed using simulation. The simulation will be run on a newly built discrete event simulation engine[1] which is able to handle the new queue mapping and the resource dependencies. This new simulator is developed in Java and makes use of the Desmo-J library. The simulation engine supports the resource dependencies, queueing strategies, case attributes and replications, warm-up periods and confidence intervals for the performance measures. A more extensive example process is available in the GitHub location to test the simulator.

The simulation model that will be used to test the hypothesis consists of two sequential activities served by the same resource type to simulate the patterns described by the resource dependencies. This provides three scenarios, where in all scenarios the same resource type is used, but which resource instance can handle the case depending on the resource dependency. Both of the activities have an exponentially distributed processing time with $\mu = 2.0$. The exponentially distributed arrival rate of the system will vary from 0.01 to the number of resources divided by the inter-arrival rate of 4.0. The number of resources in the simulation will increase from 2 to 10 resources. The duration of the simulation is 1440 h and the warm-up period is 192 h, which is visually confirmed by observing stability of the utilization rate, and 30 replications are performed per scenario. In Fig. 2 the simulation model is shown in BPMN 2.0.

In order to evaluate the effect of these dependencies the utilization rate of the simulation model is plotted against the theoretical predicted utilization rate

[1] Downloadable from: https://github.com/rmdijkman/simulator.

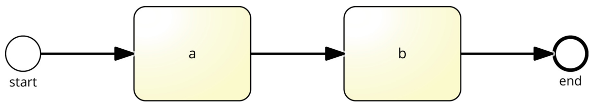

Fig. 2. Model for simulation

of the model. From Kleinrock [10] the theoretical utilization rate for an M/M/c queue is equal to: $\rho = \frac{\lambda}{c \cdot \mu}$. Where the assumption is made that the arrival process and the processing time are exponentially distributed and that there are c resources available for this queue. The theoretical expected utilization rate is adapted to correct for the number of queues which need to be served, where q stands for the number of queues which are served by the specific resource type: $\rho = \frac{\lambda \cdot q}{c \cdot \mu}$. Using this metric the theoretical utilization rate is calculated using the provided simulation parameters, assuming equal processing time for all queues. In the described simulation situation this is the case, so the metric can be used.

5.2 Results

From the results obtained in the simulation it can be concluded that the hypothesis is true, without taking the resource dependencies into account the result can differ as much as about fifty percent on performance of the resources in the process model. First the results from the separation of duties resource dependency will be discussed, next the case handling resource dependency is discussed. In each figure the three different scenario's are plotted against each other, where the blue line is the scenario without resource dependencies, the red dashed line is the scenario with separation of duties and the orange dash-dot line is the case handling scenario. For each line the 95% confidence interval is calculated and the differences are significant between the lines.

For the case where there are no resource dependencies it is observed that the achieved utilization in the simulation aligns perfectly with the theoretical utilization. This supports the claim that the simulation tool properly models the utilization of the resources.

The separation of duties resource dependency has a major impact in the scenario with only two resources as can be seen in Fig. 3. From a theoretical utilization of 0.9 there is a deviation where the resource is less optimally used in terms of utilization. For the scenario's with more resources the deviations between the separation of duties dependency and the situation without any resource dependencies are not significantly different. This can be explained by the fact that the restricting behavior of the separation of duties principle becomes less when there are more resources available. In the situation with only two resources the principle excludes 50% of the resources from handling the second task, where with five resources, see Fig. 3, only 20% of the resources is excluded from the handling the case.

The case handling resource dependency has an equal impact on the resource utilization as the separations of duties resource dependency in the scenario with only two resources, see Fig. 3, where both lines overlap but deviate from the line without resource dependencies. When increasing the number of resources the effect does not diminish as with the separation of duties resource dependency but increases significantly. In the scenario with five resources the utilization drops to 0.8 when it is expected to be 1.0. If the number of resources is further increased to eight resources then the utilization drops further to 0.7 instead of the expected 1.0. In contrast to the separations of duties resource dependency, the case handling resource dependency shows the same effect with only two resources as the separation of duties resource dependency, but in stead of a decrease in the restricting behavior the restricting behavior increases. For the scenario of two resources the principle excludes 50% of the resources from handling the second task. Where with five resources the restricting behavior has increased to excluding 80% of the resources to handle the second task. This translates to a heavily decrease of resource utilization in the model compared to the expected performance of the model. Therefore the waiting times in the model explode, since there is a lack of resources for processing these cases. Next to the lack of availability of resource because of the resource dependencies there is a random allocation strategy applied for the resources. With a process aware resource allocation algorithm this effect can be lessened, but the resource restrictions still apply, so the lack of resource availability still has an effect on the performance of the process.

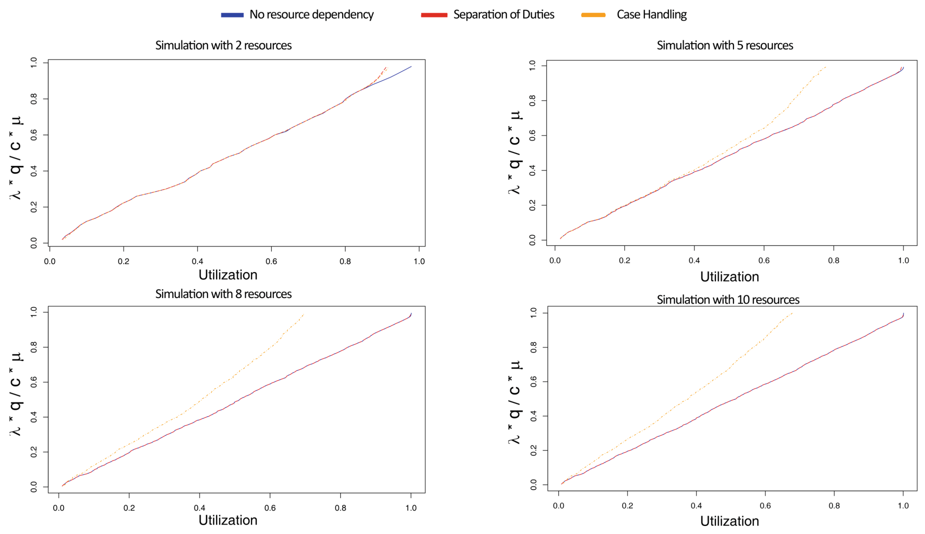

Fig. 3. Comparison with different number of resources

6 Conclusion and Future Work

This paper contributes to the field of business process management and business process simulation by providing a new simulation engine which provides the foundation for further incorporation of resource perspectives in business process simulation. In the related work section the lack of incorporation of certain aspects of business processes is shown. In the evaluation it is shown that the effects of not incorporating these aspects can lead to significant differences in performance metrics for the simulation. And since in practice companies aim for high utilization rates of 0.8 and above for their resources, these differences found are very relevant for the performance of the real world processes.

For future research the effect of the resource dependencies in larger processes can be studied to see how big the influence is on a large scale. Furthermore the use of advanced allocation algorithms for resources to cases can be investigated to obtain better performances of the model using the same set of resources. Also the resource assignment can be more formalized, as for example in Cabanillas et al. [2]. The effects of other extensions in the process model can be investigated further to show their impact on the performance of the process. For the discrete event simulation more mappings are possible, for example to map BPMN first to CPN Tools and then to discrete event simulation, here is chosen to map BPMN directly to discrete event simulation. Not all patterns from Russell et al. [18] and their interactions are described, which is a limitation of this paper. Also a limitation is that there is only looked at a single process instance, to increase the validity of the results more instances should be tested. Furthermore the simulation tool can be validated in future work by comparing it to other simulators, for example CPN Tools. In general more research should be conducted in the field of business process simulation, especially in combination with advanced resource perspectives. Thus this simulation engine is the first step in the direction of accurate simulation of complex real world processes.

Acknowledgment. The research described in this paper is supported by the Dutch Organization for Scientific Research as part of the DATAS project with grant number 438-15-507.

References

1. Andrews, T., et al.: Business process execution language for webservices (2003)
2. Cabanillas, C., Resinas, M., del-Río-Ortega, A., Ruiz-Cortés, A.: Specification and automated design-time analysis of the business process human resource perspective. Inf. Syst. **52**, 55–82 (2015)
3. Centeno, M.A.: An introduction to simulation modeling. In: Proceedings Winter Simulation Conference, pp. 15–22 (1996)
4. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
5. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, vol. 1. Springer, Heidelberg (2013)
6. Hlupic, V., Robinson, S.: Business process modeling and analysis using discrete-event simulation. In: Proceedings of the 30th WSC, pp. 1363–1370 (1998)
7. Jansen-Vullers, M.H., Netjes, M.: Business process simulation - a tool survey. In: Workshop on CPN Tools, Aarhus, Denmark, vol. 38, pp. 1–20 (2006)

8. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri Nets and CPN Tools for modeling and validation of concurrent systems. Int. J. Softw. Tools Technol. Transfer **9**(3–4), 213 (2007)
9. Kelton, W.D., Sadowski, R.P., Sturrock, D.T.: Simulation with Arena. McGrawHill, New York (2004)
10. Kleinrock, L.: Queueing Systems, volume 2. Computer Applications, vol. 66. Wiley, New York (1976)
11. Kunze, M., Weske, M.: Signavio-Oryx academic initiative. In: BPM 2010 Demonstration Track 6 (2010)
12. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis, vol. 2. McGraw-Hill, New York (1991)
13. Mahajan, P.S., Ingalls, R.G.: Evaluation methods used to detect warm-up period in steady state simulation. In: Proceedings of the 36th WSC, Winter Simulation Conference, pp. 663–671 (2004)
14. OMG: Business Process Modeling Notation (BPMN) (2011). www.omg.org/spec/BPMN/2.0/PDF
15. Recker, J.: Opportunities and constraints: the current struggle with BPMN. Bus. Process Manag. J. **16**(1), 181–201 (2010)
16. Righter, R., Shanthikumar, J.G., Yamazaki, G.: Extremal service disciplines in single-stage queueing systems. J. Appl. Probab. **27**(2), 409–416 (1990)
17. Rozinat, A., et al.: Workflow simulation for operational decision support using design, historic and state information. In: Proceedings of BPM, pp. 196–211 (2008)
18. Russell, N., Ter Hofstede, A.H., Edmond, D., Van der Aalst, W.M.P.: Workflow resource patterns. In: BETA Working Paper Series WP 127, Eindhoven University of Technology, Eindhoven, The Netherlands (2004)
19. Stroppi, L.J.R., Chiotti, O., Villarreal, P.D.: A BPMN 2.0 extension to define the resource perspective of business process models. In: XIV Congreso Iberoamericano en Software Engineering (2011)
20. Ter Hofstede, A.H., Van der Aalst, W.M.P., Adams, M., Russell, N.: Modern Business Process Automation: YAWL and Its Support Environment. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03121-2
21. Sarshar, K., Loos, P.: Comparing the control-flow of EPC and petri net from the end-user perspective. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) International Conference on Business Process Management, pp. 434–439. Springer, Heidelberg (2005). https://doi.org/10.1007/11538394_36
22. Tumay, K.: Business process simulation. In: Proceedings of the 28th Conference on Winter Simulation, pp. 93–98 (1996)
23. Van der Aalst, W.M.P., Nakatumba, J., Rozinat, A., Russell, N.: Business process simulation. In: Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1, pp. 313–338. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-00416-2_15
24. Van der Aalst, W.M.P., Ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases **14**(1), 5–51 (2003)
25. Van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. **53**(2), 129–162 (2005)
26. Van Gorp, P.M.E., Dijkman, R.M.: A visual token-based formalization of BPMN 2.0 based on in-place transformations. Inf. Softw. Technol. **55**(2), 365–394 (2013)
27. De Vreede, G.J., Verbraeck, A., Van Eijck, D.T.: Integrating the conceptualization and simulation of business processes. Simulation **79**, 43–55 (2003)

28. Wohed, P., Van der Aalst, W.M.P., Dumas, M., Ter Hofstede, A.H.M., Russell, N.: Pattern-based analysis of BPMN (2005)
29. Zur Muehlen, M., Recker, J.: How much language is enough? In: Seminal Contributions to Information Systems Engineering, pp. 429–443. Springer, Heidelberg (2013)