Resource-aware Business Process Simulation: An Approach Based on Workflow Resource Patterns and ExtendSim

ABSTRACT

Simulation of business processes is one of the widely-used approaches supporting organization's decisions and planning changes. One of the limitations of existing process-aware simulation approaches is the poor support for the resource perspective. Without proper modeling of human resource behavior as well as constraints on that behavior, the results of the simulation studies will be incorrect and misleading.

This paper is taking a first step towards a resource-aware business process simulation. We are building on the well-known workflow resource patterns and model them in process simulation models. First contribution is that we refine the relationship among those patterns to modularize their realization in either a simulation or an enactment environment. Second contribution is realization of those patterns in ExtendSim, a general purpose simulation tool. We evaluate our approach on a sample scenario and report the results compared to resource-ignorant approaches.

CCS Concepts

•**Computer systems organization** **Embedded systems;** *Redundancy;* Robotics; •**Networks** Network reliability;

Keywords

Business process simulation; workflow resources patterns; Resources perspective; ExtendSim

SAC 2018,April 09-13, 2018, Pau, France

DOI: DOI:http://dx.doi.org/10.1145/2851613.2851735

1. INTRODUCTION

Business process simulation (BPS) involves performing simulation experiments based on accurate models, reflecting the business process behavior, with respect to performance metrics such as cycle time, cost and resources utilization [3, 25]. Simulation of business process is essential to help organizations plan their needs, not just from IT-infrastructure, but also from resources and manpower. In order to have an effective process simulation, business experts, who are not necessarily simulation language fluent must be able to prepare and run simulations. This, in our view, can be achieved by making complexities of simulation models as transparent as possible to the user. Simulation setup should be integrated with process modeling tools and running in the backend in the same way currently soundness, e.g. Signavio [1]. In a simulation view, the user has to specify all necessary parameters, e.g., activities execution times, cases arrival rate, etc. Moreover, the user has to be able to specify the different resources with their properties and distribution among roles as well as constraints on resource selection and assignment to perform activities. All these data have to be merged and transformed into a simulation model of some target language, run and results are reported back to the user on the process view. This operation is iterative by nature where the user can change any of the simulation parameters, the process model, resources etc and rerun the simulation.

There are several challenges to realize this view. The first challenge is to effectively enforce resource selection and assignment constraints in simulation models. The second challenge is to find realistic simulation parameters for the model. The third challenge is to find a simulation tool that would require the least effort of modeling as well as providing rich reports. The fourth challenge is to make the translation from process models, e.g. XML files of process definitions to the simulation language as transparent as possible.

The second challenge has been partially addressed in literature [12] where process logs can be used to obtain realistic values for parameters like task durations, case arrival rate, branching probability etc. Also the challenge has been addressed in [16] where event logs were used to discover knowledge about human resources assignment to activities. The third challenge has also been partially addressed by the approaches in [2, 1] where process-specific simulation tools are built rather than building upon readily available general pur-

[1]https://www.signavio.com/

pose simulation tools. Respectively, the fourth challenge is waived out in such case. The few approaches available for business process-aware simulation, e.g.[2, 1] provide primitive or no support for modeling the resource perspective in their simulation models. In most of the cases, human resources are represented as number of items within a specific role. Without proper modeling of resource activities, simulation results are misleading and ineffective. Thus, the first challenge is still out standing. That is, proper modeling of human resources selection and assignment to tasks. It is obvious that without proper modeling of the perspective, simulation results are misleading.

The contributions in this paper addresses the first challenge. We take a first step towards human-resource-aware process simulation. We provide a systematic approach that uses the well-known workflow resource patterns [13] to model constraint on resource availability, selection and assignment at process simulation time. The first contribution in this paper is refining both intra and interrelationships among those resource patterns. This helps modularize the modeling and realization of those patterns. The second contribution is to model those patterns in a general purpose simulation tool, ExtendSim [5]. Then, we compare the resource-aware simulation models with other process simulation tools and different what-if scenarios.

The rest of the paper is organized as the following: Section 2 revisits the workflow patterns and further refines their relationships in order to modularize their use in simulation or process execution. Section 3 shows how resource patterns are modeled in ExtendSim. Evaluation of our approach and comparison with related approaches are given in Section 4. Section 5 discusses the related work. Section 6 concludes the paper with an outlook on future work.

2. REVISITING RESOURCES PATTERNS

According to [13], resources patterns are categorized into seven groups: creation patterns for resources selection during design time, push patterns for allocating work items to resources, pull patterns where resources can pick work items, detour patterns specifying work items delegation among resources, auto-start patterns where work items are triggered by specific event, visibility pattern determining resources visibility for work items and multiple-resource patterns which are concerned with tasks that require more than one resource working on it concurrently.

Following a divide-and-concur approach, in this paper, we are concerned with the first two groups, namely the creation and push patterns. We believe that building simulation models where those patterns can be employed is a first step towards getting more accurate simulation results. Pull, detour and multiple-resource patterns are currently out of scope and subject for future research. Visibility patterns are purely related to process enactment and building of process execution engines, similar to implementation of work item lists. Thus they are not considered for process simulation. Similarly, auto-start patterns are not considered for resource-aware simulation as they are related to automatic tasks.

Creation patterns are concerned with *which resource are eligible?* whereas push patterns are concerned with *How to pick one of the eligible resources?*. Creation patterns lend themselves to resource selection at design time. That is out of the of all available resources R, a creation pattern cp is responsible for finding set R_{cp} R which represents the candidate resources where any of them is capable of executing the respective task t. R_{cp} can be either specified by properties that each resource r R_{cp} must possess in order to be able to execute t or it can be specified by explicitly enumerating its members. On the other hand, push patterns are more on the execution or simulation time assignment of a work item wi to a resource r R_{cp} where wi is the instance of task t within a specific process instance. So, the enforcement of a push pattern should result in at most one specific resource being assigned to the work item whereas enforcement of a creation pattern results in a set of candidate resources R_{cp}. Note that R_{cp} might be empty in case of none of the available resources possesses sufficient capabilities to perform t.

It is helpful to study the relationships between patterns to build a minimal set of constructs that can be reused in the related patterns. Previous studies [13] about such relationships introduced a simple "related to" relationship between pairs of patterns either across or within the same pattern category. Another study [15], presented a process mining approach based on resource-aware the textual Declarative Process Intermediate Language (DPIL) to model organizational (resources) perspective.

Here, we refine these relationships between creation and push patterns using *complement-perspective* where push patterns complement creation patterns. For instance, if the creation pattern *Direct Distribution* is specified, then it does not make sense to allow *Random Allocation* as a push pattern as the resource set is already a singleton. Rather, *Distribution by Offer Single Resource* or *Early Distribution* are much more suitable. Table 1 summarizes these complements relationships.

2.1 Creation Patterns Intra Relationships

The intra-relationships among creation patterns are summarized in Figure 1. They are: the *is-a* relationship, the *complements* relationship, the *opposite to* relationship to indicate that patterns are the opposite or negation of each other, precautions have to take place when those opposite patterns are applied on an overlapping set of tasks. We have three patterns at the top of the hierarchy: Role-based Distribution, Authorization and Retain Familiar. The other creation patterns are seen as special cases of one of those patterns as follows:

Direct distribution can be modeled as role-based distribution by making sure that only one resource is a member of the role. ***Deferred distribution*** is a special case of role-based distribution through specifying the role at design time but the actual members will be resolved at run (simulation) time. However, the resolution logic at runtime has to be specified in the simulation model. this can fall back to other distribution patterns. ***Capability-based distribution*** is a special case of role-based as a virtual role can collect as members all resources that possess those capabilities. The capability constraints are evaluated against the whole set

Table 1: Complement relation between creation and push patterns

Creation Patterns	Complement (used with) Push Patterns
Direct Distribution	Distribution by Offer - Single Resource Distribution by Allocation - Single Resource Early Distribution
Role-based Distribution Organizational Distribution	Distribution by Offer - Multiple Resources Random Allocation Round Robin Allocation Shortest Queue
Deferred Distribution	Late Distribution
Authorization	Distribution by Offer - Multiple Resources Random Allocation Round Robin Allocation Shortest Queue
Separation of Duties	Distribution by Offer - Single Resource Distribution by Offer - Multiple Resources Distribution by Allocation - Single Resource Random Allocation Round Robin Allocation Shortest Queue
Case Handling	Distribution by Allocation - Single Resource
Retain Familiar	Distribution by Allocation - Single Resource
Capability-based Distribution History-based Distribution	Distribution by Offer - Multiple Resources Random Allocation Round Robin Allocation Shortest Queue

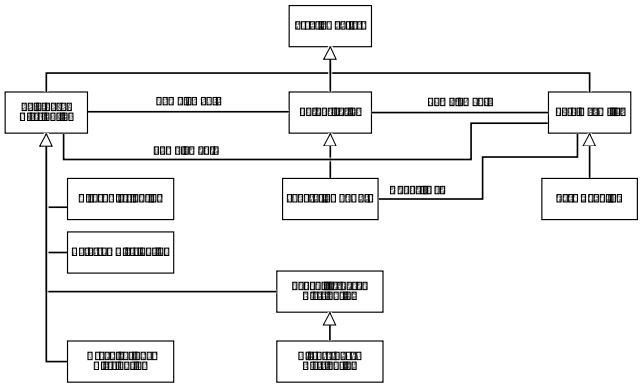

Figure 1: Creation patterns Intra-relationships

of resources and matching individuals are added to the virtual role. This step can take place either before running the simulation, in the form of preparing simulation data or can be done as one of the initialization steps of the simulation model. In the latter case, the selection logic is part of the simulation model and is run prior to the very first task in the process. ***History-based distribution*** is a special case of capability-based distribution where we can add a property `Number of completed items` along with its value to the set of capabilities to look for. The value shall be supplied as part of the simulation parameters. ***Organizational distribution*** is special case of role-based distribution because we can simulate the organizational position as a role. ***Authorization pattern*** according to [14] touches both design- and run-time aspects of accessibility of a resource to a work item. Looking from the design time perspective, authorization adds a security framework that complements the work item distribution to resources. That is why in Figure 1 we put it as complementary to both role-based distribution and retain familiar patterns as it is possible to use it to refine the allocation. From the runtime perspective authorization is concerned with the actions a resource can take with respect to work items in his list like starting, suspending, delegating etc. For simulation purposes, we are concerned with the design time perspective and we model it in the same way we model the role-based distribution pattern. ***Separation of duties pattern*** is considered as a special case of the authorization pattern. Our argument here is that separation of duty states that the performer of task B cannot be the same performer of task A within the same case. In that sense, at runtime performer of task A in a given case is *not authorized* to perform task B for the same case. Separation of duties is also considered *opposite to* ***retain familiar pattern*** where in retain familiar the resource is executing the whole case while in separation of duties resource is allowed to execute part of the case depending on some specification. Thus, it is a contradiction to have both patterns applied on the same process model. ***Case handling pattern*** is a special case of retain familiar where all work items of the case are distributed to the same resource.

2.2 Push Patterns Intra Relationships

Push patterns consists of nine patterns subdivided into three sets as in Figure 2. The work distribution set is concerned with the offering of work task to a single or multiple re-

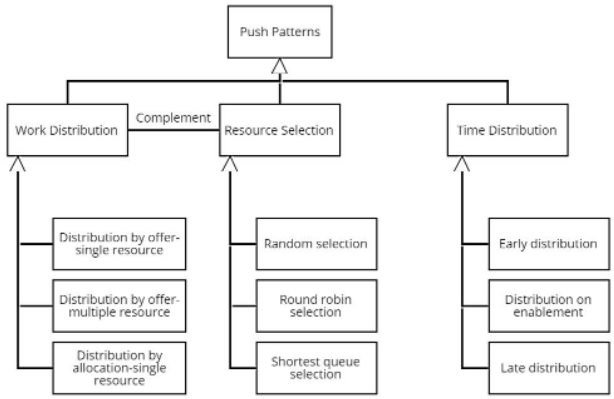

Figure 2: Push patterns Categorization

sources and allocating a task to a single resource. The resource selection set identifies a suitable resource when many possible candidates are available and allocate work item based on random allocation, round robin or shortest queue which is complement to be used with work distribution set. Finally the time distribution set spots the time of task distribution to an appropriate resource. The early distribution indicates the ability of resource to view future work list. This is irrelevant to simulation similar to the visibility resource patterns [22], distribution on enablement is the default distribution mechanism for resources over tasks while late distribution would lead to blocking the running simulation experiment as task would wait for the required resource undefined time period accordingly early and late distribution patterns were excluded from our modeling demonstration.

3. EXTENDSIM

ExtendSim [5, 17] is a powerful, general-purpose simulation tool for developing dynamic models representing current or proposed real-life business processes using building blocks allowing the adjustment of assumptions to arrive at an optimum simulation experiment. We chose ExtendSim for several reasons, 1) a general-purpose simulation tool 2) extensible to compose new constructs or to tweak the behavior of existing ones, to fit resource-awareness simulation needs 3) able to communicate with third-party applications as the ultimate goal is to make the simulation runs transparently to the end user. However, the realization of the resource patterns and their relationships as discussed in Section 2 can be produced in other (general) purpose simulation tools, probably with varying degrees of efforts.

Figure 3 describes ExtendSim constructs that we used to model resource patterns.

3.1 Patterns Representation in ExtendSim

In this section, we show how the creation and push patterns are modeled in ExtendSim. Namely, we discuss in details the role-based distribution, separation of duties and retain familiar creation patterns, cf. Figure 1. We also discuss changes needed in those models to address patterns specifications such as handling resource capabilities in *capability-based distribution*. For each general pattern, we show an excerpt of the ExtendSim model and explain how work items (cases) flow in and how they are (*batched*) bound to and (*unbatched*) unbound from a resource. Before discussing patterns modeling, Section 3.1.1 describes how resource elements, pools, items are prepared in a simulation run.

3.1.1 Resources Preparation

Out of the box, ExtendSim supports both direct allocation and role-based allocation by means of resource items or resource pools constructs. Resources are generally represented as numbers and there is no built-in means to distinguish resources. This is a limitation when it comes to modeling patterns like capability or history-based distribution.

Our approach to model resources assumes that we have a single resource item where each resource is annotated with relevant properties for the simulation. At minimum, each resource is annotated with 1) the role he is assigned in the organization, 2) identifier, this is assigned at the beginning of the simulation run, 3) his workload, as the number of items assigned to the resource and awaiting execution. The resource identifier is relevant for patterns like retain familiar, case handling and separation of duty. The workload property is relevant for implementing shortest queue push patterns.

Other properties could be, e.g., years of experience, certificates, number of completed cases with respect to the simulation model in hand, etc. Our main assumption here is that each resource is a member of at most one role.

During the resources-preparation step as in Figure 4, each ExtendSim model must contain a number of resource items constructs equivalent to the number of roles involved in the simulation. Then, the first part of the ExtendSim model is to distribute those resources from the single source to the respective resource item. ExtendSim has a so-called database construct in which, relational tables can be built, linked and later-on queried by other elements in a model. We use this feature to define all relevant resources along with their properties in the database. To read the resources from the database and distribute them among resource items, an initial *resource item* block is added and the total number of resources is specified. Next, a *read* block is used to read the required records from the database. The initial (left most) resource item block is actually redundant but it is a limitation by ExtendSim[2]. Each read record from the database is passed to an equation block where the routing logic to the respective resource item block is implemented, based on the resource's role. Actual routing takes place following *select item out* block which finally places the resource in its respective pool. One can think of the equation block as the place to set values for properties and the select item out block as an *if, else if, else* block.

Based on our discussion of resource patterns in Section 2, recall that creation patterns are concerned with selecting eligible resources for the respective work item (task), whereas

[2]Read block in ExtendSim needs a trigger to access records in the database. So, we make the resource item as input so that there will be as many reads from the database as the number specified in the resource item block.

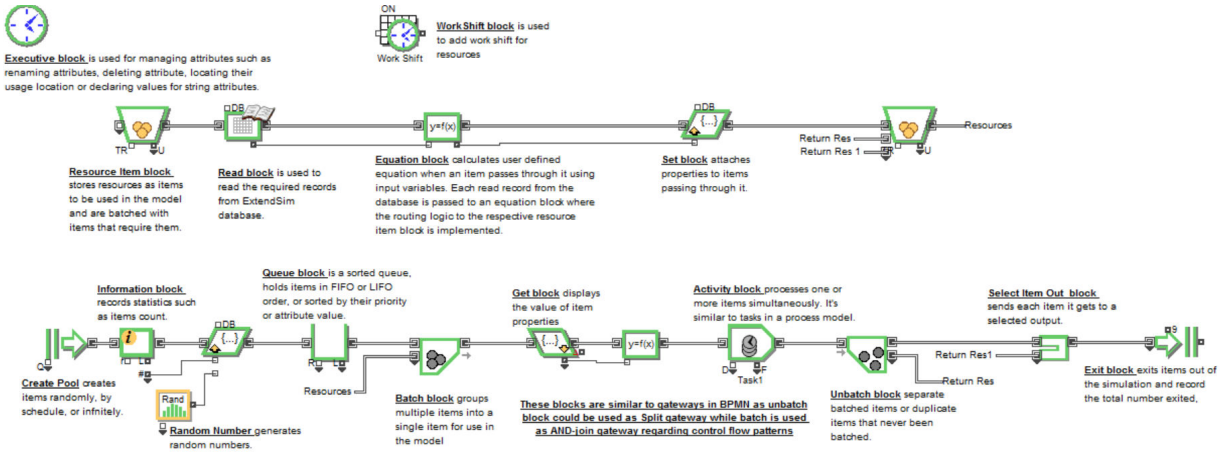

Figure 3: ExtendSim building blocks

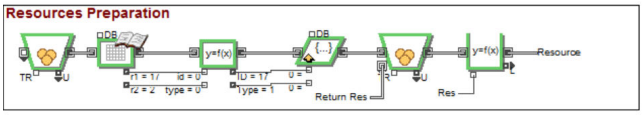

Figure 4: Template of preparing resources for offering and execution in ExtendSim

push patterns are concerned with assigning the work item to exactly one of those candidates. Thus, it is crucial to reflect that in the simulation. That is, a resource eligibility to execute a work item should not be affected by the situation where this resource might be busy executing other work items. Thus, the resource pools are duplicated for each separate role. One pool is used for resource selection while the other is used for assignment. However, such duplication in resource preparation step is not affecting simulation time as these steps occur before cases arrival (starting simulation model). The default selection of resources in ExtendSim is based on round robin push pattern.

3.1.2 *Modeling Creation and Push Patterns*

Role-based distribution with Push Patterns. The role-based creation pattern is modeled with four push patterns. The first modeled push pattern is *distribution by offer to multiple resources* as shown in Figure 5.

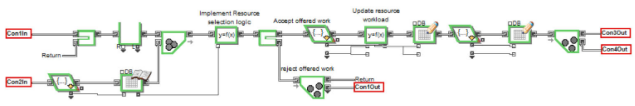

Figure 5: Template to model role-based offering to multiple resources in ExtendSim

Cases(work items) arrive at arrival *queue* block. The work item is offered to a resource using a *batch* block and the workload value of the resource is retrieved from database using *read* block. If the resource accepts the work item then an *equation* block is used to increase the resource's workload by one and updates the database record using *write* block. Also, the work item executer property is set to the resource's identifier and is saved in the database. After that the resource is unbatched from the work item to make him available for resource selection (other creation patterns) for other tasks. If the resource rejects the *offered* work item, it is sent back to the waiting queue to be offered to another resource.

By default, offering work items occur by round robin push pattern. But it is applicable to randomly select resources or select based on resource working queue(pick resource with shortest queue, i.e., least workload). For random selection of a resource, we specify a uniform integer distribution for resources priority and select from the resources waiting queue based on that priority and the shortest queue pattern is modeled using the *queue tool* value block and workload attribute is defined for all resources specifying the length of the work items queue for every resource.

As mentioned in Section 2.2, offering work items is based on *distribution on enablement* push pattern, thus offering logic is run when the execution of work items is enabled. The execution of work items is shown in Figure 6. Work items are selected from the waiting queue and the executer ID is determined to select the appropriate resource from the execution resource items. If the executer is unavailable the work item is sent to the unavailable resource waiting queue. There, the work item waits for a specific time interval before being entered to execution again. This repeats until eventually the designated resource is available or the simulation run ends. After task execution, the workload of the executer is decreased by one and all relevant database records are updated. This execution logic is independent from how the resource was selected, i.e., the creation pattern implemented.

Modeling special cases of role-based distribution, e.g. *organizational distribution*, *capability-based* , etc., see Figure 1, is realized by changing the body of the equation block labeled *implement selection logic* in Figure 5. For instance, *organizational based distribution* is implemented by matching on the organizational position property.

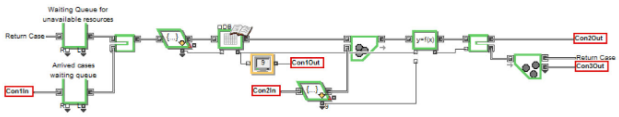

Figure 6: Template to model work items execution in ExtendSim

Retain familiar with Push Patterns. Retain familiar is an n-ary pattern that needs to check the assignment of resources over a set of tasks. Essentially, execution needs to remember the resource who executed the first task in that group within a case and make sure that all other tasks in the same case are assigned to the same resource that is why this pattern is complemented with *distribution by allocation-single resource* push pattern. Figure 7 shows how to model retain familiar allocation to single resource in ExtendSim. The *read* block is used to retrieve the required resource ID who undertook a preceding work item.. the *queue* and *write* block are used to allocate the work item to required resource.

Figure 7: Retain familiar distribution pattern in ExtendSim

Case handling distribution is a special case of retain familiar pattern, the model segment of matching the resource identifier is thus repeated before every task in the model.

Separation of duties is opposite to *retain familiar* pattern. Thus, it can be modeled in the same way as in 7 for distribution by allocation-single resource where the only difference is the body of the equation block which changes from equality to inequality check. Modeling other complement push patterns with separation of duties is similar to role based offering in 5 with changing the selection equation to inequality.

Authorization distribution can be viewed from three angles. The first is related to privileges that a resource needs to possess in order to be able to execute a task. The second view is with respect to what a resource can do with a running task such as suspend, restart, abort etc. the third view is with respect to how many work items a resource can activate concurrently.

Resource privileges can be checked in the same way role-based distribution is implemented. That is, extra attributes are added to the resource and are checked in the respective equation block. The second view is related to the concept of a task lifecycle model. That is, the different states a task instance can assume during a running case. In our model, we have a simple lifecycle model which assumes that tasks will be completed if activated. Our focus in this paper is more on modeling resource selection. However, such extension is possible based on [6]. The third view is more related to pull patterns and is out of scope.

4. EVALUATION

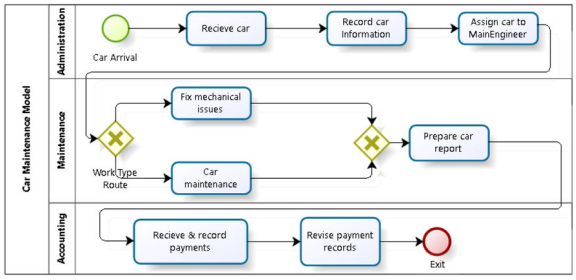

Figure 8: Car maintenance process modeled in BPMN

In this section we evaluate the correctness of our approach through two methods. The first method is to compare our approach with Bizagi Modeler [2] and BIMP [3] through BIMP online simulator [4]. The reason for choosing Bizagi and Signavio among other BPS tools is that both are process-specific simulation tools. Both tools are limited with respect to resource-awareness. They basically support role based distribution. However, the main purpose of this comparison is a notion of validity check for our work. That is for the simple case of role based distribution, our approach gives similar or close results to those tools.

The second method of evaluation is building a simulation model where several constraints are enforced and different what-if scenarios are examined with respect to the number of resources in each role and where patterns other than role-based distribution are considered. In the latter method, we run the simulation model on ExtendSim alone as Bizagi and signavio do not support resource allocation beyond role-based distribution. The creation patterns for the second evaluation method are *role based, separation of duties, retain familiar* and *capability based distribution*. Push patterns modeled are *distribution by offer-multiple resources, distribution by allocation-single resource* and *shortest queue*. Due to space limitations, several details have been omitted. Yet, they can be found through this link: https://db.tt/3fANDbKVr2

4.1 Comparison with Bizagi Modeler and BIMP

We modeled the process in Figure 8 with our approach in ExtendSim, Bizagi and BIMP. The purpose of this comparison is to check the accuracy of the modeling of our modeling approach and not comparing the functionality of the tools. Only role-based distribution is modeled in that example. We have fixed all simulation parameters for all implementations.

The designed model is simulated for 8 hours with case arrival interval of 10 minutes and total number of process insances equals 49. The processing time of each task is constant. We chose constant time so we can compare results of the simulation on Bizagi Modeler, ExtendSim and BIMP. The model is explaining a car maintenance center including the administration , car maintenance and accounting tasks. For each of the three roles, cf. Figure 8, we assigned 3 admins, costing

[3]https://www.signavio.com/
[4]http://bimp.cs.ut.ee/

$20/hr each, 6 maintenance engineers, costing $50/hr each, and 3 accountants, costing $30/hr each.

Results. We compared the number of cases arrived, number of cases completed, the average execution time of each task, resource utilization and total cost in Figure 9. The number of started and completed instances are the same in Bizagi and ExtendSim models. BIMP forces adding the total number of process instance to complete as input parameter in addition to the desired simulation run length. Thus the number of started instances equals the competed instances. That leads to a variation in the simulation results specially the total waiting time as shown in Figure 9. The

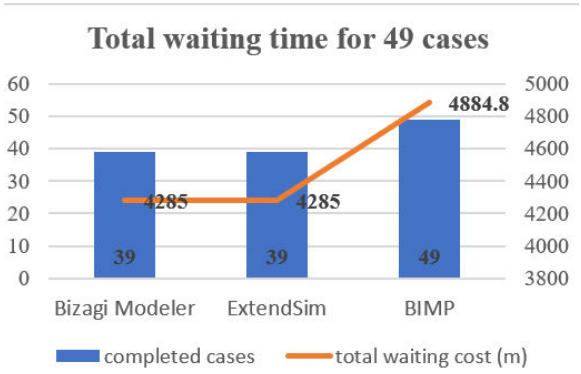

Figure 9: Comparing total waiting time(m) in Bizagi, ExtendSim and BIMP

minimum, maximum and average processing time are identical as tasks are assigned constant execution time. There were deviations in the results of total time due the number of instances completed per task in the BIMP models. Resource utilization and cost in both ExtendSim and Bizagi models are the same for admins and accountants roles while maintenance engineer results have a small deviation between Bizagi and ExtendSim models due to the number of completed instances in each. BIMP shows different results for resources utlilzation and total cost due to the number of completed tasks. Overall, results are almost identical for ExtendSim and Bizagi models. We interpret that as an indicator of the correctness of our modeling approach. All detailed results can be found following the link indicated above.

4.2 What If Scenarios

Here, we extend the model in Figure 8 by enforcing role based, retain familiar, capability-based and separation of duties patterns. The model was run 5 times with different number of resources.

The first run contains 4 resources (1 admin, 1 maint. eng. and 2 accountants), the second run contains 6 resources (2 for each role), the third run contains 12 resources (4 for each role), the fourth run contains 24 resources (8 of each role) and the fifth run contains 32 resources (8 admins, 8 accountants and 16 maint. eng.). We used two accountants in the first run because tasks depending on accountants initiate separation of duties pattern that requires minimum two resources. If the number of used resources are less than minimum (one resource) the model will stop running. For the fifth run, we increased only the maintenance engineer resource as the, other two resources (admins and accountants) were able to handle all started instances with no waiting time.

Role based distribution pattern is modeled on admin and accountant resources and tasks *receive car* and *receive & record payments*. Retain familiar pattern is applied on tasks *record car information* and *assign car to MainEngineer*. Capability-based pattern is applied on maintenance engineer resources and tasks *Fix mechanical issues* and *car maintenance*. Separation of duties is enforced on accountant resources and the two tasks *receive & record payments* and *revise payment records*.

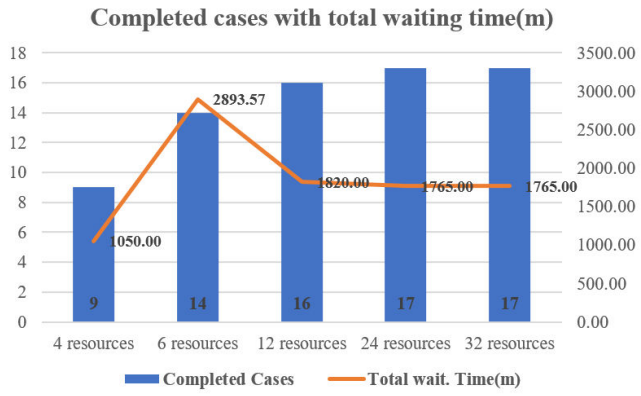

Figure 10: Completed cases with total waiting time(m)

Figure 10 compares the total number of resources in each run with total waiting time, number of started and completed instances respectively. The total waiting time decreases as number of resources used increases until waiting time reaches zero minutes at a total number of 24 resources. Analogously, the number of completed instances increases as number of resources increases as well. We can see that at a total number of 24 resources the maximum of 32 cases can be handled a day. This is true as increasing the number of resources to 32 did not contribute any gain in the number of completed instances.

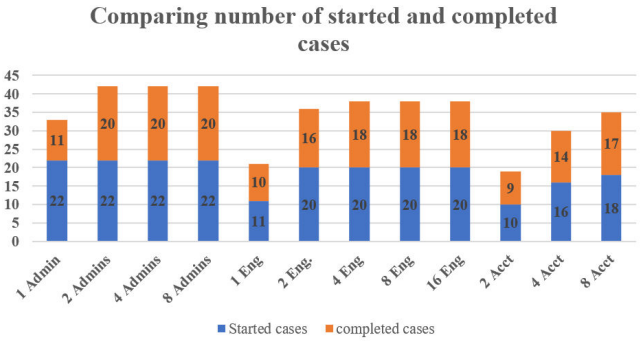

Figure 11: Roles with di←erent capacities vs. started and completed cases

In Figure 11, the different resources' roles are compared relating the number of started and completed instances.

Analyzing the above graphs leads to configuring the optimal number of resources required to reduce the waiting cost without utilizing insufficient number of resources.

5. RELATED WORK

Several approaches, e.g.[18, 4, 10], discuss the need for business process simulation tools as integral part of business process management.

[19, 21] discuss the limitations of business process simulation usage in reality and categorize the risks of current simulation settings. These pitfalls were further extended and refined in [20]. Amongst such risks is the inadequate modeling of resource behavior and availability. Our work is contributing to the avoidance of this risk by means of pattern-based enforcement of resource behavior constraints. Moreover, in [21], the authors use CPN tools by example to address the pitfalls identified where they only addressed modeling of resource availability without constraining resource allocation as we discuss in this paper.

A simulation environment based on YAWL workflow tool and CPN tools was discussed in [25] allowing experiment to start from an intermediate stage rather than empty execution state. Resources were only examined related to availability, utilization and count. A hybrid approach for business process modeling and simulation with limited resources modeling related to concurrency aspects was addressed in [23]. The approach examined concurrent execution of tasks through resources multitasking to avoid blocking resources for long time. Our work can be seen as complementary to both approaches. However, the concurrent execution by human resources is of very limited use especially when it comes to knowledge intensive tasks.

The work in [9] discusses supporting business process simulation model construction by using event logs. They provide means to extract Resources behavior from logs. Modeling resource behavior is suppoerted by roles, schedules, handling procedures, resources unavailability and grouping resources based on similar activities handling. Yet, constraints on resource behavior was not identified by the authors. Thus, our work complements theirs in that regard.

A survey on business process simulation tools was conducted in [7]. Amongst the evaluation criteria was the ability to model the resources perspective. Amongst the evaluated tools were Arena and CPN tools. The study results indicated that CPN tools provides better support for resources representation than Arena. However, both require quite an effort to model resources behavior in a way that might be inaccessible to business process experts, see our discussion about tool selection in Section 3.

[24] presents L-Sim, a Java based simulation engine to simulate BPMN diagrams. L-Sim provides resources capabilities such as assigning human and non-human resources to tasks, the ability to pick tasks based on FIFO or priorities, preempt resources, specifying multiple resources to the same task and specifying alternative available resource. Again, constraints on resources behavior is not supported. However, L-Sim support a subset of pull and push patterns. OX-ProS - an Open and Extensible Process Simulator for BPMN was discussed in [3], the tool converts BPMN to Colored Petri-Nets (CPN). The tool supports modeling role-based and chained execution distribution in workflow resources patterns. Other resource patterns were left for future work. A blueprint architecture was proposed in [8] for a business process simulation engine that focuses on BPMN models. Resources modeling was only concerned with resource availability.

In [11], modeling separation of duty constraints for BPEL4People was discussed. The focus was on an executable language rather than on simulation purposes. Yet, the paper discusses variants of separation of duty constraints which form interesting future directions to model those variants at simulation time.

6. CONCLUSION AND FUTURE WORK

We have taken a first step towards resource-aware process simulation by modeling workflow resource creation and push patterns in ExtendSim. We have evaluated our approach against other process-specific simulation tools and showed that results are similar for commonly supported scenarios. Moreover, we evaluated different what-if scenarios that were guided by the total number of completed cases in the simulation run.

So far, resources were modeled as passive elements. In the future, we aim at modeling pull workflow patterns in which resources can actively select what work items to execute. Moreover, we aim at supporting complex task lifecycles where a work item can be delegated, canceled, failed etc.

7. REFERENCES

[1] BIMP. *BIMP [Computer software]*, 2017.

[2] Bizagi. *Bizagi Modeler [Computer software]*, version 3.1.0.011 edition, 2016.

[3] L. Garcia-Banuelos and M. Dumas. Towards an open and extensible business process simulation engine. In *CPN Workshop*, 2009.

[4] V. Hlupic and S. Robinson. Business process modelling and analysis using discrete-event simulation. In *Winter Simulation Conference*, pages 1363–1370. IEEE Computer Society Press, 1998.

[5] Imagine That Inc. *ExtendSim [Computer Software]*, version 8 edition, 2010.

[6] M. Jansen-Vullers, R. IJpelaar, and M. Loosschilder. Workflow patterns modelled in arena. Technical report, Eindhoven University of Technology, 2006.

[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*, volume 38, pages 1–20, 2006.

[8] S. Krumnow, M. Weidlich, and R. Molle. Architecture blueprint for a business process simulation engine. In *EMISA*, volume 172, pages 9–23, 2010.

[9] N. Martin, B. Depaire, and A. Caris. The use of process mining in business process simulation model construction. *Business & Information Systems Engineering*, 58(1):73–87, 2016.

[10] B. Mathew and R. Mansharamani. Simulating business processes-a review of tools and techniques. *International Journal of Modeling and Optimization*, 2(4):417, 2012.

[11] J. Mendling, K. Ploesser, and M. Strembeck. Specifying separation of duty constraints in bpel4people processes. In *BIS*, pages 273–284. Springer, 2008.

[12] J. Nakatumba, M. Westergaard, and W. M. P. van der Aalst. Generating event logs with workload-dependent speeds from simulation models. In *CAiSE Workshops*, volume 112 of *LNBIP*, pages 383–397. Springer, 2012.

[13] N. Russell, W. M. van der Aalst, A. H. Ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In *CAISE*, volume 3520 of *LNCS*, pages 216–232. Springer, 2005.

[14] N. Russell, W. M. van van der Aalst, and A. H. M. ter Hofstede. *Workflow Patterns: The Definitive Guide*. The MIT Press, 2016.

[15] S. Schonig, C. Cabanillas, S. Jablonski, and J. Mendling. Mining the organisational perspective in agile business processes. In *International Conference on Enterprise, Business-Process and Information Systems Modeling*, volume 214 of *LNBIP*, pages 37–52. Springer, 2015.

[16] S. Schonig, C. Cabanillas, S. Jablonski, and J. Mendling. A framework for efficiently mining the organisational perspective of business processes. *Decision Support Systems*, 89:87–97, 2016.

[17] J. Strickland. *Discrete event simulation using ExtendSim 8*. Lulu. com, 2013.

[18] K. Tumay. Business process simulation. In *Winter Simulation*, pages 93–98. IEEE Computer Society, 1996.

[19] W. M. van der Aalst. Business process simulation revisited. In *Workshop on Enterprise and Organizational Modeling and Simulation*, pages 1–14. Springer, 2010.

[20] W. M. Van Der Aalst. Business process simulation survival guide. In *Handbook on Business Process Management 1*, pages 337–370. Springer, 2015.

[21] W. M. Van der Aalst, J. Nakatumba, A. Rozinat, and N. Russell. Business process simulation. In *Handbook on Business Process Management 1*, pages 313–338. Springer Berlin Heidelberg, 2010.

[22] W. M. Van der Aalst, M. Weske, and D. Grunbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.

[23] O. Vasilecas, A. Smaižys, and A. Rima. Business process modelling and simulation: hybrid method for concurrency aspect modelling. *Baltic Journal of Modern Computing*, 1(3-4):228–243, 2013.

[24] A. Waller, M. Clark, and L. Enstone. L-sim: Simulating bpmn diagrams with a purpose built engine. In *Winter Simulation Conference*, pages 591–597. IEEE, 2006.

[25] M. T. Wynn, M. Dumas, C. J. Fidge, A. H. Ter Hofstede, and W. M. Van Der Aalst. Business process simulation for operational decision support. In *BPM*, pages 66–77. Springer, 2007.