Design of an Extensible BPMN Process Simulator

Chapter · January 2018

DOI: 10.1007/978-3-319-74030-0_62

3 authors, including:

Luise Pufahl
Technische Universität München
61 PUBLICATIONS **594** CITATIONS
SEE PROFILE

Mathias Weske
Hasso Plattner Institute
401 PUBLICATIONS **14,414** CITATIONS
SEE PROFILE

Design of an Extensible BPMN Process Simulator

Luise Pufahl, Tsun Yin Wong and Mathias Weske

Hasso Plattner Institute, University of Potsdam
{Luise.Pufahl,Mathias.Weske}@hpi.de, TsunYin.Wong@student.hpi.de

Abstract. Business process simulation is an important means for quantitative analysis of a business process and to compare different process alternatives. With the Business Process Model and Notation (BPMN) being the state-of-the-art language for the graphical representation of business processes, many existing process simulators support already the simulation of BPMN diagrams. However, they do not provide well-defined interfaces to integrate new concepts in the simulation environment. In this work, we present the design and architecture of a proof-of-concept implementation of an open and extensible BPMN process simulator. It also supports the simulation of multiple BPMN processes at a time and relies on the building blocks of the well-founded discrete event simulation. The extensibility is assured by a plug-in concept. Its feasibility is demonstrated by extensions supporting new BPMN concepts, such as the simulation of business rule activities referencing decision models and batch activities.

Key words: Business process simulation, extensibility, BPMN

1 Introduction

Business process management (BPM) has been introduced by organizations to make their daily business operations efficient and flexible with respect to their execution environment. For this purpose, business processes are captured by process models with a process modeling language. The resulting process models are used as blueprint for process execution [19]. Once a process is documented, it can be used for simulation to identify its most optimized form [5]. Business process simulation (BPS) is a quantitative analysis technique which provides insights into throughput times, resource utilization and process costs of different process alternatives [4]. It is a cost-effective way to gain insights into an existing or a future situation of the business process execution [1].

The most common process modeling language in BPM practice as well as research is the Business Process Model and Notation (BPMN) which also includes semantics for an automated process execution [11]. Some BPM tool vendors already provide BPMN simulators, e.g., Bizagi Modeler, BonitaSoft, Visual Paradigm, and Trisotech Modeler [5], for which a translation of the BPMN process diagram in a specific simulation language is not necessary anymore. This avoids errors in the pre- and post-phase of process simulation and eases the usability.

BPS is also used by researchers to evaluate new process modeling artifacts. It eases the communication of the value of new ideas to the BPM community. However, commercial BPMN simulators of tool vendors and academic products, such as BIMP [2], are mainly proprietary and do not support the extensibility by new BPMN constructs.

Thus, CPN Tools [15] relying on colored petri nets (CPNs) is mainly applied for process simulation in research [8]. However, it requires a manual translation of the BPMN process diagrams into CPNs and expert knowledge to work with the tool. Further, existing simulators do not support the simulation of multiple processes which run concurrently and compete for the same resources. This is relevant in order to observe the resource utilization in case of several business processes.

In this paper, we present the design and architecture of a proof-of-concept implementation for an extensible BPMN process simulator which also considers the simulation of multiple concurrent business process models. It relies on the building blocks of computer simulation as well as on aspects of existing BPS software. The architecture requires a framework for discrete event simulation (DES). With DES, a real-world process is captured as a finite set of events in time, i.e., each event occurs at a certain point in time and marks the change of the process state [3]. As no changes occur between events, the simulation jumps from one event to another, allowing DES to run fast and independently from real process time in contrast to continuous systems. Tumay [17] describes DES as the "most powerful and realistic tool for analyzing the performance of business processes", which provides "statistical input and output capabilities and advanced modeling elements [...]." The extensibility mechanism of the simulator is based on a designed plug-in structure which provides well-defined entry points into the simulation environment. Additionally, the extensibility has the advantage that BPMN features can be separated in mandatory and optional ones. By outsourcing certain BPMN features in a plug-in, the performance of simulator can be increased, because only those BPMN features are used which are necessary for the respective simulation use case. The feasibility of the plug-in structure is demonstrated based on two recently introduced BPM constructs: (1) business rule activities referencing decision models [12], and (2) batch activities [14].

The remainder of the paper is structured as follows: in Section 2 the related work, specifically other academic simulation prototypes and architectures are discussed. Based on the background in Section 3 about BPS and DES, the design and architecture of the extensible BPMN process simulator is presented in Section 4. Thereby, we discuss the functionality of the different components, the mapping of BPMN constructs into DES, the selection of mandatory BPMN constructs from a simulation perspective, and which are provided by plug-ins, and, finally, the plug-in structure. The feasibility of plug-in structure is demonstrated in Section 5 by extending the simulator with features of current BPM research. The paper concludes in Section 6.

2 Related Work

Business process simulation (BPS) software can be differentiated into various categories: (1) BPM tools extended with a simulation functionality (e.g., Bizagi Modeler, Trisotech Modeler), (2) stand-alone BPMN simulation tools (e.g., BIMP), and (3) general purpose simulation tools (e.g., CPN, Arena being a discrete event simulator). Several survey exists on the evaluation of different BPS tools, for instance, by Jansen-Vullers and Netjes [8], or by Freitas and Pereira [5] which is specifically on BPMN simulators. However, they do not discuss the general extensibility of the BPS software

tools. Existing commercial tools are proprietary and do not provide any extension mechanism; therefore, we focus in the following on architectures and prototypes developed in the academic context:

Wynn et al. [21] present a procedure for BPS with the input and output data of each step. Based on observation of historical data from a process execution engine, the simulation model is instantiated with values referring to a certain point of time in execution, allowing short-term simulation of business processes. The approach is evaluated by using the YAWL workflow engine and CPN tools. This work focuses on the connection between a process execution engine and a simulator; extensibility is not discussed and the BPS is done based on YAWL models.

In [6], García-Bañuelos and Dumas describe an open and extensible business process simulator which transforms a BPMN process diagram into a hierarchical colored petri net (CPN) which is then simulated by CPN tools. The transformation is based on templates describing for each BPMN construct how to map it into CPN fragments. The templates can be adapted or extended for new BPMN constructs. However, the transformation requires very detailed and technical CPNs in order to represent the resource handling and corresponding timing aspects [8]. This leads to a high overhead for developers and needs a profound understanding of CPNs. A similar approach can be found by Krumnow et al. [10] presenting an architectural blueprint for a BPMN simulator in which BPMN concepts are first mapped into a formal language, petri nets, which are easily translatable into DES concepts. In this work, extensibility of the simulator is not discussed . Petri nets can replay the control flow of a business process, but they are constrained in executing specific activity behavior, for instance, simulating a service call.

In contrast, Rücker developed a BPS solution [16] based on the DES software DESMO-J [7]. At the time of development, it was integrated in the open source business process engine JBoss jBPM[1] and supported models in the proprietary language jPDL. Today, it allows the simulation of BPMN models, but it is not open source. It is integrated in the commercial JBoss BPM Suite[2]. In contrast, Wagner et al. [18] suggest to modify DES specification itself to represent process activities. The authors present a conceptual model in which activities are represented as a complex DES event having a start event and an end event, and optionally a resource assigned. However, its detail level is not sufficient in comparison to the BPMN specification [11]. For example, the enablement of an activity, which can be different to its start, is missing.

The BIMP simulator developed in the work of Abel [2] is a academic process simulator which has been optimized for performance. This simulator does not base on any existing simulation framework and is not open for extensions. In summary, most of the presented simulation tools lack to support extensibility and multiple concurrently running business processes using the same set of resources. We address these aspects after having introduced some basics on BPS and DES in the next section.

[1] `https://www.jbpm.org/` [accessed 23 May 2017]

[2] `http://developers.redhat.com/products/bpmsuite/overview/` [accessed 23 May 2017]

3 Background

This section serves as brief introduction into business process simulation in Section 3.1 and discrete event simulation in Section 3.2, which are the conceptual basis for the extensible BPMN process simulator.

3.1 Business Process Simulation

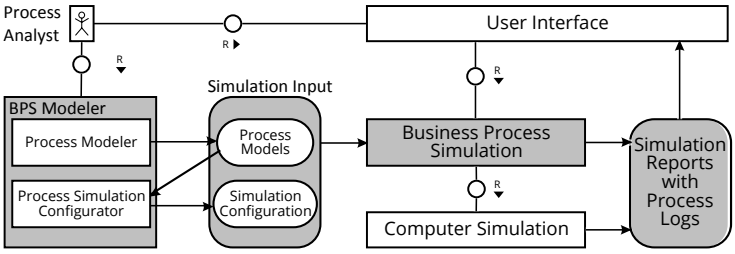

Fig. 1. Architecture of a business process simulator (as FMC block diagram [9]).

To explain the functionality of business process simulators, Fig. 1 provides a generalized architecture. The goal of a business process simulator is to imitate the execution of a number of process instances – different process executions – based on the given simulation input to generate an artificial history [4]. This artificial history delivers information about the probable throughput time, process costs and resource utilization of the current process design. The simulation input consists of the process model, the resource information (resource types, number of resources, timetable etc.) and the stochastic information about the process, such as the arrival rate with its associated distribution, the distribution of the activity duration, and the branch probability [21]. Stochastic information about the process can be either estimated by process experts or identified based on the analysis of process execution logs. The *simulation input* is provided by the *process analyst* with the help of a *BPS modeler* providing means to model the process and to provide all relevant information for the simulation. The process modeling functionality is not always part of a business process simulator. When the process analysts starts a simulation run, the *Business Process Simulation* parses the simulation input, sets up the simulation experiment, and runs it for the defined number of instances [21]. In case a general computer simulation tool is used for the simulation, e.g., CPN tools or a DES simulator, the BPMN constructs of the process diagram have to be translated into the concepts of the *Computer Simulation*. When a simulation run is finished, the artificial process logs are generated. They are enriched with resource and cost information. Based on them, simulation reports with statistics on the flow time, process costs, resource utilization etc. can be generated which are presented to the process analyst.

3.2 Discrete Event Simulation

In contrast to continuous simulation in which the system dynamics are tracked over time, in a discrete event simulation (DES) the behavior of a complex system is mapped

to a discrete sequence of well-defined events [3]. An event marks a specific change of the system and it is happening at specific instants in time. The simulation can jump from event to event as no state change occurs between two consecutive events which makes the simulation time efficient.

Different frameworks exist that guide the creation of simulation models in terms of time and state, the so-called world views, e.g., the event scheduling approach, where each event contains a routine which decides on the next events to be scheduled, or the activity scanning approach, where activities are due by condition. The activities define a fixed incrementation of simulation time. When an activity finishes, the conditions of all other activities are checked. In addition to the primary world views, several modifications exist, such as the three-phase simulation approach [13]. It is based on the activity scanning approach where each point of time is simulated, leading to slow execution. This drawback is addressed by complementing it with features of the event scheduling.

For our business process simulator, we will use the DES framework DESMO-J[3] which natively supports the process interaction approach and the event scheduling approach. The framework is open for extensions, thus enabling the modification of the event scheduling approach to use the three-phase simulation approach. DESMO-J has not only gained wide acceptance in the academic community; it is also the foundation of many commercial simulation software. It provides blueprints for simulation models in which discrete events and entities can be defined. A simulation model collects and references resources which are required throughout the simulation, defines the built-in logging configuration, and defines the first steps of a simulation, for instance, the initialization of discrete events and the scheduling of the first event. A discrete event is defined by an event routine. It is optionally assigned to an entity which can be modified by the event. To start a simulation, the simulation model is connected to a DESMO-J experimentation component, optionally with a temporal end condition. This "black-box" component executes the first steps defined in the simulation model and is responsible to monitor, log, and terminate the simulation.

4 Design and Architecture of the BPMN Process Simulator

After having introduced the basics of BPS and DES, this section introduces the design and architecture of an extensible BPMN process simulator. First, the architecture is sketched in Section 4.1 and the functionality of the different components is described. Next, the mapping of the BPMN constructs into DES is presented in Section 4.2. Thereby, the coverage of the BPMN constructs by the simulator is discussed. Here, we differentiate between BPMN constructs being covered by the basic simulator and those being captured as plug-in to ease the performance of the simulator. Finally, the plug-in concept is presented in Section 4.3 to allow a flexible extensibility of the simulator.

4.1 Architecture

The architecture of the extensible simulator for business process simulation is shown in Fig. 2. To focus on the internal behavior of the process simulator, we limited the architecture regarding the interaction of the simulator with its environment to the mandatory aspects.

Simulation Manager. The *Process Analyst* accesses the *Business Process Simulator* via the *Simulation Controller* and selects the process models and configurations for a simulation run. The selected simulation input is loaded by the *Simulation Manager* which accesses all other components of the process simulator. The *Configurations & Process Model Parser* validates the input and prepares it for translation by the *Simulation Model Translator & Instantiator*. The latter builds the process-specific DES simulation model which is then called by the DES experiment running the simulation. From the insights collected by the Simulation Manager, the *Output Logger* creates *Log Files* which are the output of the simulation.

Simulation Input. As resources can be involved in multiple processes of their organization, the resources are defined in the *Global Configuration*. In this, the type of resources, the number of resources, and their time tables are described; concrete resources can be defined, too. Summarized, the input for the process simulator consists of an arbitrary number of *BPMN Process Diagrams*, a *Simulation Configuration* for each process diagram, in which the arrival rate distribution, the activity duration distribution, the branching probability etc. is given, and one *Global Configuration* file for the organizational details. To ease extensibility, all configuration files are provided in an open format, such as XML.

Configurations & Process Model Parser. This component consists of several parsers to convert each BPMN process diagram and its simulation configuration as well as the global configuration into structures which can be efficiently queried during simulation.

[3] `http://desmoj.sourceforge.net/`

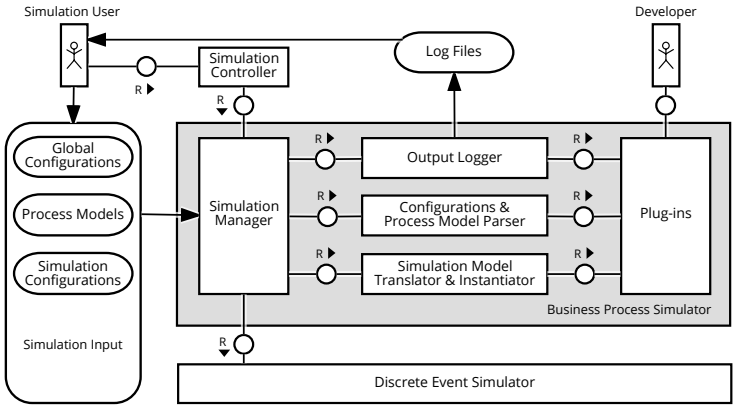

Fig. 2. Architecture of prototypical implementation of the extensible BPMN process simulator (as FMC block diagram [9]).

These structures are designed in a way, such that they are able to cover the openness of the format used for the configuration files.

Simulation Model Translator & Instantiator. This component is responsible to translate the simulation input into a DES simulation model and to initialize the DES experiment executing the simulation model. Therefore, first, the parsed input is translated into static model components like queues, distributions and structures for data collection. Second, the initial state of the simulation model is instantiated. The initial events, e.g., a event representing the generation of process instance, for the event list of the *Discrete Event Simulator*, are created and activated.

Output Logger. The simulator supports low-level DES logging as provided by the *Discrete Event Simulator* to debug its implementation. For providing a simulation output, high-level logging is supported by capturing process-relevant events (e.g., the enabling of an activity, the start of a new process instance). This is used to finally create the artificial process logs and a simulation report including several calculated key performance indicators (KPIs), such as the average throughput time.

Plug-Ins. Regarding the extensibility, well-defined entry points are desired for the architecture of the process simulator. We provide the concept of *Plug-Ins*, allowing the development of new functionality for the process simulator. A plug-in refers to one single feature and can be switched on and off before simulation, thus supporting modularity on simulation usage level. The plug-in concept will be presented in Section 4.3.

Details on the actual prototypical implementation of the extensible BPMN process simulator called *Scylla*[4] can be found in [20].

4.2 Mapping of BPMN constructs into a Discrete-Event Simulation

After having parsed the simulation inputs and represented them in an internal structure, each identified BPMN construct is translated into a DES construct in order to design the DES simulation model. BPMN constructs which represent an occurrence at a point in time, i.e., start, intermediate and end events, and gateways are mapped directly to discrete events. The routines of the discrete events are responsible for selection and scheduling of the next discrete event based on the configurations of the simulation model. As BPMN tasks are executed over a period of time, their states are mapped to a set discrete events according to the BPMN activity lifecycle [11] with each routine of the event describing the transition from the state. The specific type of each of these constructs is represented in the discrete event by attributes. The events process an entity which represents a process instance. During simulation, the entity stores information on the resources used and events to be scheduled at a later point in time.

If a simulation experiment is started, a discrete event generates a process instance for each process model by creating and scheduling the event which represents the BPMN start event of the process model. To simulate more process instances, it reschedules itself for a later point in time according to the arrival rate, which is defined in the simulation model configurations. The discrete event representing the BPMN start event

[4] Its source code is available at https://github.com/bptlab/scylla.

immediately executes its event routine as described in the previous paragraph. The simulation experiment terminates by either a time constraint defined in the configuration or if all process instances have been completed.

After presenting the mapping, we want to continue with discussing the coverage of BPMN constructs. A plug-in concept does not only support new developers to implement features without modifying the system, it also allows to reduce the size of the simulator by moving optional parts to plug-ins; hence easing the performance of the system. Therefore, it is necessary to define a minimum set of BPMN elements which we consider as sufficient to conduct a BPS. In case of BPMN constructs which are not part of the minimum set, the core process simulator is designed in a way that it will not stop its simulation; instead, the simulator will skip or replaces them with supported constructs of similar type. Further, BPMN constructs have to be identified which are very relevant for process simulations and should be supported by plug-ins.

As a result, the standard representations of the BPMN constructs *Flow*, *Task*, *Start Event*, *End Event* are part of the minimum set. Since these constructs are located in *Pools* and *Lanes*, these two are supported as well. They refer to task roles and to the idea of resource allocation, which is essential in BPS. Furthermore, gateways are part of the minimum set, because they are responsible for creating, collecting, and directing tokens in the execution of a process model. An exception is the exclusive gateway which should be available as a plug-in to allow different versions of the branch selection. It can be either driven by stochastic distributions or by the values of data objects.

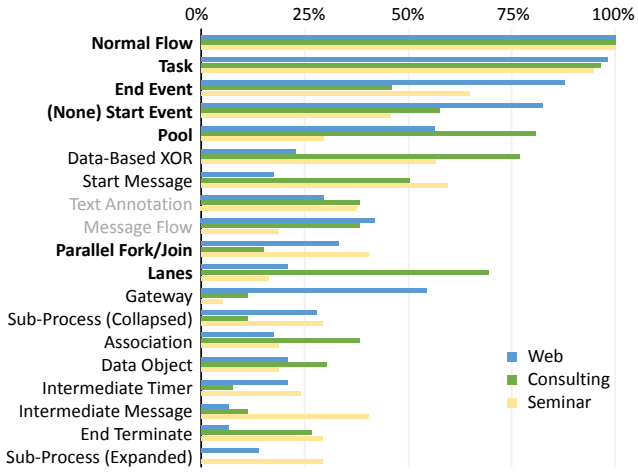

Fig. 3. List of frequently used BPMN constructs and their support in the business process simulator, cf. [23].

Additional BPMN constructs to be supported by plug-ins are selected based on their usage frequency by practitioners which has been thoroughly investigated by zur Muehlen and Recker [23]. Fig. 3 shows a list of the most frequent BPMN constructs, descending order sorted by their coverage in the diagrams [23]. Constructs considered for the core process simulator are written in bold, whereas constructs to be supported in plug-ins are not emphasized. Here, we want to highlight that, in contrast to existing BPS, also a plug-in is provided where also the values of data objects can be simulated. Additionally, constructs which are not supported are displayed in gray, e.g., *Text Annotations* having no influence on the process execution or *Message Flow* which requires an interaction with another organization.

4.3 Plug-In Structure

In software engineering, extensibility of a system is crucial for third-party developers aiming at implementing new features. Different extensibility mechanisms can be differentiated [22]: Whereas an *open-box* system allows the direct modification of the source code, a *glass-box* system allows the extension of the open source code without changing it. A system is considered as *black-box*, if system details are completely hidden and extensions can only be added by a finite set of mechanism. Extensibility of the BPMN process simulator is allowed by a *grey-box* approach where a list of abstractions is provided which can be refined by plug-in developers like in the black-box approach, but it also provides access to the source code like in the glass-box approach. Thereby, we want to provide a plug-in structure with a high degree of extensibility which cover the complete simulation flow from input parsing to output logging.

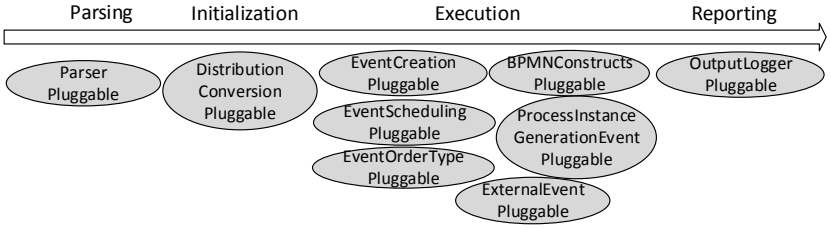

Fig. 4. Abstract classes as entry points for writing plug-ins categorized into the different steps of a simulation.

As result, several abstract classes are defined as entry points for developers to write plug-ins shown in Fig. 4 in which they are categorized into the different stages of simulation: parsing, initialization, execution, and reporting. The *ParserPluggable* offers an entry point to the input parsers, such that new simulation input, e.g., a new BPMN element, can be parsed. During initialization of the DES simulation model, the DESMO-J distribution for the arrival rate and activity distributions are set. The entry point *DistributionConversionPluggable* allows the initialization of additional distributions. While executing a simulation experiment, events are generated, stored in queues, and if an event occurs, their event routines are executed which usually results in new events. Different entry points are available to influence DES events: *EventCreationPluggable* to generate new type of events, *EventSchedulingPluggable* to influence the scheduling of events, and *EventOrderTypePluggable* to adapt the priorities of events and changing their order in the queues. For influencing the implemented BPMN behavior of the simulator, two entry points exists: one on the process instance level – *ProcessInstanceGenerationEventPluggable*– and one on the BPMN events level – *BPMNConstructsPluggable*. The latter ones includes several sub-classes to influence the behavior of the minimum set of BPMN elements supported by the basic simulator (cf. the BPMN constructs written in bold in Fig. 3), for instance, *BPMNStartEventPluggable*, *TaskEnableEventPluggable*. The *ExternalEventPluggable* offers the opportunity to add behavior which is not strictly related to a single process instance, but to the general behavior of business process simulation. Finally, a plug-in entry point, the *OutputLoggerPluggable*, is available to extend the simulator logs and output reports.

Our implementation [20] contains plug-ins for several advanced BPMN constructs, for the logging of simulated process executions and for the calculation of their KPIs.

5 Demonstration of Feasibility

In this section, we discuss the feasibility of the plug-in structure by integrating new plug-ins for two new types of BPMN activities: One the hand to enable the simulation of decision-aware activities (i.e., business rule tasks) based on DMN (Decision Model and Notation) models [12] and on the other hand the simulation of batch activities which allow batch processing in business processes:

Simulation of DMN Models: With the introduction of the DMN standard, a possibility of modeling complex operational decisions was created. The DMN standard complements the BPMN standard and leads to decision-aware business processes. In current BPS, decision models are ignored. Therefore, we extended the simulator to allow the simulation of business rule tasks referencing DMN decision models. For this, two plug-ins were necessary: On the one hand, the *DMNTaskBeginPlugIn* was created which extends the *TaskBeginPluggable* being one of the *BPMNConstructsPluggables*. In case of a business rule task, it uses the input data of the currently simulated process instance and the referenced decision rules to call a rule engine. The output of the rule engine is then written into a output data object of the business rule task. This output can be later used, e.g., for the branching at the exclusive gateway, then being driven by data. On the other hand, a *DMNStatisticLoggerPlugIn* extending the *OutputLoggerPluggable* was created which documents how often which output was returned and which branch was then selected with its associated throughput time and connected costs. The plug-in structure allows a quick way to integrate a new activity behavior by adapting the event routine of the task-begin-event. Further, the access to the output logger allows to generate specific simulation reports.

Simulation of Batch Processing: Recent works, e.g., in [14], have integrated so-called *batch activities* into business processes to allow batch processing. Batch processing enables a business process, which usually act on a single item, to bundle the execution of groups of process instances for particular activities in order to improve its performance. Incorporating batch activities in the BPMN simulator requires an extension with regards to all steps of a simulation: parsing of a new type of BPMN element, changing the execution to allow the collection of process instances and their synchronized execution, and logging of batch-specific KPIs (e.g., costs reduction or waiting time due to batch processing). For realizing a batch activity, five entry points were used. First of all, the *BatchParserPlugIn* extending the *ParserPluggable* was created to parse the extension elements used for a batch activity consisting of different batch configuration parameters, such as the batch activation rule. For a batch activity, the challenge is that the normal activity behavior has to be adapted. Activity instances are interrupted in their execution and only if a certain condition is fulfilled, a batch of instances is executed. Therefore, the *BatchTaskEnablePlugIn* extending the *TaskEnablePluggable* assures that enabled batch activity instances are assigned to a batch cluster after activity enablement. In case a new batch cluster has to be created, a new type of event – a *BatchClusterStartEvent*

– is prepared. It implements the blueprint of a DESMO-J discrete event with the batch cluster as its entity. Its event routine schedules all task-begin-event objects of the instances in the batch cluster and selects one process instance which represents the batch execution of the others and is executed as usual. For the remaining instances, the *BatchTaskBeginPlugIn* ensures that the task-begin-event routine is not executed, instead their task-terminate-events are scheduled. After the representative instance was executed, the *BatchTaskTerminatePlugIn* allows that the task-terminate-events of the others are activated and that resulted logging data of the representative ones is taken over by the others. Finally, the *BatchLogger* extending the *OutputLoggerPluggable* creates a report with batch-specific KPIs, such as reduced costs, the maximum and average waiting time of the batch clusters. With the existing plug-in structure, it was possible to integrate a new activity type which also changed the execution semantics. Thereby, the main challenge was after interrupting the process instance execution to return to the normal execution. This could be solved by adapting the scheduling of events using the *TaskEnable-*, *TaskBegin-*, and *TaskTerminatePluggable*.

6 Conclusion

In business process management, simulation has gained acceptance as a tool for quantitative analysis of business processes. This paper showed that existing BPMN simulators are limited in their extensibility. Therefore, the design and architecture was presented for an extensible BPMN process simulator to support also academic researchers in estimating the impact of new concepts on the process execution. From a set of well-founded computer simulation paradigms, discrete event simulation was chosen as the basis of our simulator since the event-based behavior maps directly to the tokenized behavior in process execution. As opposed to existing simulation software, it also supports simulation models consisting of multiple business processes which may be run independently from each other.

The presented simulator enables researchers with programming knowledge to create plug-ins. Plug-ins extend the behavior of the simulator at well-defined interface points which are distributed over all core components, i.e. input parsing and instantiation, simulation experimentation and output logging. Furthermore, they provide modularity on development and usage level and offer separation between mandatory and optional features. The feasibility of the architecture was evaluated by extending the simulator with new BPMN concepts. The extensions cover the full range of pluggables. It showed that the plug-in structure allowed an easy and structured means for integrating new types of BPMN activities including parsing and generation of special simulation output reports. However, further extensions will show whether the current design of plug-in interfaces is complete or need further extensions. We plan, for example, to integrate a resource pluggable with which other resource allocation mechanism (currently the direct and role allocation is supported) can be integrated. Additionally, a plug-in may work properly when used alone, but the interaction between certain plug-ins may cause problems. In future, we plan to evaluate the performance of the simulator with its plug-in concept.

References

1. Van der Aalst, W.M., Nakatumba, J., Rozinat, A., Russell, N.: Business process simulation. In: Handbook on Business Process Management 1, pp. 313–338. Springer (2010)
2. Abel, M.: Lightning Fast Business Process Simulator. Master's thesis, Institute of Computer Science, University of Tartu (2011)
3. Banks, J.: Discrete-event system simulation. Pearson Education India (1984)
4. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of business process management, vol. 1. Springer (2013)
5. Freitas, A.P., Pereira, J.L.M.: Process simulation support in bpm tools: The case of bpmn. In: 5th International Conference on Business Sustainability. 2100 Projects (2015)
6. Garcıa-Banuelos, L., Dumas, M.: Towards an open and extensible business process simulation engine. In: CPN Workshop 2009 (2009)
7. Göbel, J., Joschko, P., Koors, A., Page, B.: The discrete event simulation framework DESMO-J. In: ECMS. pp. 100–109 (2013)
8. Jansen-Vullers, M., Netjes, M.: Business process simulation–a tool survey. In: 7th Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (2006)
9. Knöpfel, A., Gröne, B., Tabeling, P.: Fundamental Modeling Concepts: Effective Communication of IT Systems. Wiley (2005)
10. Krumnow, S., Weidlich, M., Molle, R.: Architecture blueprint for a business process simulation engine. In: EMISA. vol. 172, pp. 9–23 (2010)
11. OMG: Business Process Model and Notation (BPMN), Version 2.0 (January 2011)
12. OMG: Decision Model and Notation (DMN), Version 1.1 (June 2016)
13. Pidd, M., Cassel, R.A.: Three phase simulation in java. In: Proceedings of the 30th conference on Winter simulation. pp. 367–372. IEEE Computer Society Press (1998)
14. Pufahl, L., Meyer, A., Weske, M.: Batch regions: process instance synchronization based on data. In: 18th International Enterprise Distributed Object Computing Conference (EDOC). pp. 150–159. IEEE (2014)
15. Ratzer, A.V., Wells, L., Lassen, H.M., Laursen, M., Qvortrup, J.F., Stissing, M.S., Westergaard, M., Christensen, S., Jensen, K.: Cpn tools for editing, simulating, and analysing coloured petri nets. In: International Conference on Application and Theory of Petri Nets. pp. 450–462. Springer (2003)
16. Rücker, B.: Building an open source Business Process Simulation tool with JBoss jBPM. Ba thesis, Stuttgart University of Applied Science (2008)
17. Tumay, K.: Business process simulation. In: Proceedings of the 27th conference on Winter simulation. pp. 55–60. IEEE Computer Society (1995)
18. Wagner, G., Nicolae, O., Werner, J.: Extending discrete event simulation by adding an activity concept for business process modeling and simulation. In: 2009 Winter Simulation Conference (WSC). pp. 2951–2962. Winter Simulation Conference (2009)
19. Weske, M.: Business process management: concepts, languages, architectures. Springer (2010)
20. Wong, T.Y.: Extensible BPMN Process Simulator. Master's thesis, Hasso Plattner Institut, University of Potsdam (2017)
21. Wynn, M.T., Dumas, M., Fidge, C.J., Ter Hofstede, A.H., Van Der Aalst, W.M.: Business process simulation for operational decision support. In: International Conference on Business Process Management. pp. 66–77. Springer (2007)
22. Zenger, M.: Programming language abstractions for extensible software components. Ph.D. thesis, École polytechnique fédérale de Lausanne (2004)
23. Zur Muehlen, M., Recker, J.: How much language is enough? theoretical and practical use of the business process modeling notation. In: International Conference on Advanced Information Systems Engineering. pp. 465–479. Springer (2008)