Extensible BPMN Process Simulator

Luise Pufahl and Mathias Weske

Hasso Plattner Institute at the University of Potsdam, Germany
{Luise.Pufahl,Mathias.Weske}@hpi.uni-potsdam.de

Abstract. Business process simulation is an important means for quantitative analysis of a business process. With the Business Process Model and Notation (BPMN) being the state-of-the-art language for the graphical representation of business processes, a variety of process simulators exist which already support the simulation of BPMN process diagrams. However, they do not provide well-defined interfaces to integrate new concepts into the simulation environment. This work demonstrates an open and extensible BPMN process simulator written in Java with defined entry points for extensions based on a plug-in structure. The demo presents the architecture of the simulator and presents how it can be extended. It is aimed at researchers in the business process management field who want to evaluate new modeling artifacts by simulation.

Keywords: Business process simulation, extensibility, BPMN

1 Introduction

Business process simulation (BPS) as a quantitative analysis form for business processes is a cost-effective way to get insights into the throughput times, costs, resource utilization etc. of different business process designs [2]. The goal of a business process simulator is to imitate the execution of a number of process instances – different process executions – based on a given simulation input, such as a process model, stochastic information on activity durations etc., and the resource information, to generate artificial logs as basis for the different process statistics. With Business Process Model and Notation (BPMN) becoming an industry-standard, widely used in research and practice, different BPMN simulators exists, e.g., Bizagi Modeler, BonitaSoft, Visual Paradigm, and Trisotech Modeler [3], 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 modeling artifacts. However, commercial BPMN simulators of tool vendors and also academic products, such as BIMP [1], are mainly proprietary and do not support the extensibility to new BPMN constructs.

In this demo, we present an open and extensible BPMN process simulator which builds up on a discrete simulation software called Desmo-J[1]. It provides a plug-in structure with different entry points into the simulator which covers the complete simulation flow from input parsing to output logging. The current version of the simulator consists

[1] http://desmoj.sourceforge.net/

of several plug-ins for advanced BPMN concept. Further, plug-ins for a new BPMN concept, batch activities [5] which allow batch processing in business processes. This is used to evaluate the functionality of batch activities. Additionally, the simulator is able to simulate several process models using same resources in one simulation run.

In the remainder, we present the basic architecture and functionality of the simulator in Section 2. Then, the plug-in structure is presented in Section 3 in which we will also describe existing plug-ins and how a new plug-in can be added. In Section 4, we draw a short conclusion.

2 Tool Architecture and Implementation

The current architecture of the extensible BPMN process simulator, which is written in Java, is shown in Fig. 1 as FMC block diagram [4] in which data stores are represented as ellipses and active components reading and writing in data stores as rectangles.

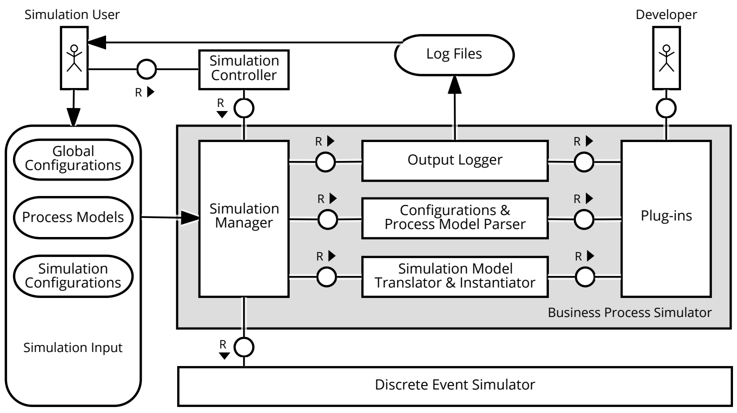
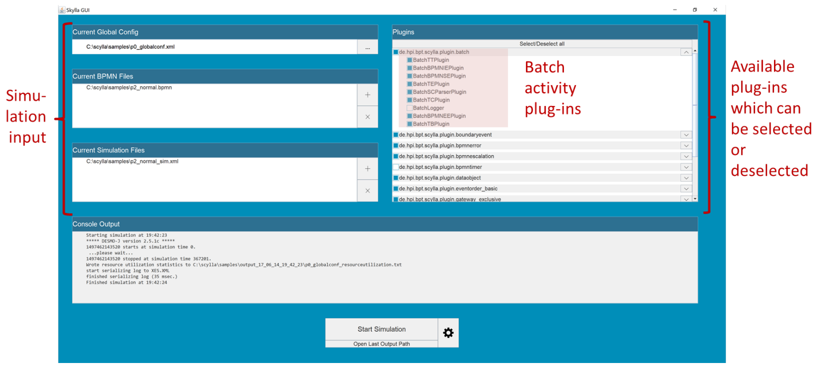


Fig. 1. Architecture of the extensible BPMN process simulator.

The *Process Analyst* accesses the **BPMN Process Simulator** via the **Simulation Controller** – the current user interface of the simulator shown in Fig. 2. Here, a user of the simulator can select the simulation input and can configure a simulation run. As the simulator allows simulation of multiple process models at the same time, the **simulation input** can consists of an arbitrary number of *BPMN Process Diagrams* and a *Simulation Configuration* for each process diagram, in which the arrival rate distribution, the activity duration distribution, the branching probability etc. is given (cf. Fig. 2). As resources can be involved in multiple processes of their organization, the resources are defined in one *Global Configuration File*, where the type of resources, the number of resources, and their time tables are described; concrete resources can be defined, too. To ease extensibility, all configuration files are provided in XML.

The selected simulation input is loaded by the **Simulation Manager** which accesses all other components of the process simulator. First, it calls the **Configurations & Process Model Parser** consisting of several parsers to convert the different simulation input files into internal structures which can be efficiently queried during the simulation. With this, the **Simulation Model Translator & Instantiator** is called which 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. Initial events, e.g., an event representing the generation of process instance, for the event list of the *Discrete Event Simulator*, are created and activated.



Current Global Config
C:\scylla\samples\p0_globalconf.xml

Current BPMN Files
C:\scylla\samples\p2_normal.bpmn

Current Simulation Files
C:\scylla\samples\p2_normal_sim.xml

Plugins
Select/Deselect all
- ☑ de.hpi.bpt.scylla.plugin.batch
  - ☑ BatchTTPlugin
  - ☑ BatchBPMNIEPlugin
  - ☑ BatchBPMNSEPlugin
  - ☑ BatchTEPlugin
  - ☑ BatchSCParserPlugin
  - ☑ BatchTCPlugin
  - ☐ BatchLogger
  - ☑ BatchBPMNEEPlugin
  - ☑ BatchTBPlugin
- ☑ de.hpi.bpt.scylla.plugin.boundaryevent
- ☑ de.hpi.bpt.scylla.plugin.bpmnerror
- ☑ de.hpi.bpt.scylla.plugin.bpmnescalation
- ☐ de.hpi.bpt.scylla.plugin.bpmntimer
- ☑ de.hpi.bpt.scylla.plugin.dataobject
- ☑ de.hpi.bpt.scylla.plugin.eventorder_basic
- ☑ de.hpi.bpt.scylla.plugin.gateway_exclusive

Console Output
```
Starting simulation at 19:42:23
***** DESMO-J version 2.5.1c *****
1497462143520 starts at simulation time 0.
 ...please wait...
1497462143520 stopped at simulation time 367201.
Wrote resource utilization statistics to C:\scylla\samples\output_17_06_14_19_42_23\p0_globalconf_resourceutilization.txt
start serializing log to XES.XML
finished serializing log (35 msec.)
Finished simulation at 19:42:24
```

Fig. 2. User Interface of the BPMN process simulator.

From the insights collected by the Simulation Manager, the **Output Logger** creates the artificial process logs and a simulation report including several calculated key performance indicators (KPIs), such as the average throughput time. 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 in the **Simulation Controller** (cf. Fig. 2) before simulation, thus supporting modularity on simulation usage level. By switching a plug-in on, its code is additionally executed at the defined entry point. The plug-in concept will be presented in next section.

Details on the actual prototypical implementation of the extensible BPMN process simulator called *Scylla*[2] can be found in [6].

3 Plug-In Concept

The demonstrated process simulator is open-source, but provides also a list of abstractions which can be refined to extend the process simulator with new concepts.

Therefore, several abstract classes are defined as entry points for developers to write plug-ins. In Fig. 3, 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

[2] Its source code is available at `https://github.com/bptlab/scylla` and a screen cast is available at `https://bpt.hpi.uni-potsdam.de/Public/scylla`.

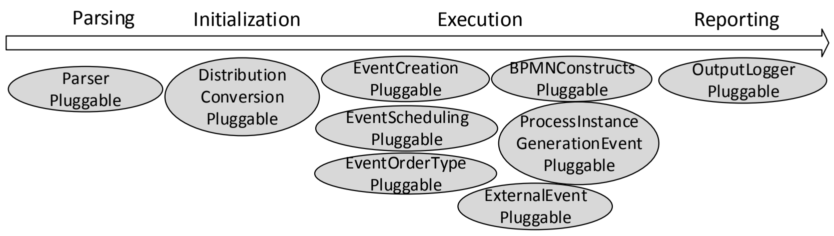


Fig. 3. Abstract classes as entry points for writing plug-ins categorized into the different steps of a simulation.

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, 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.

Fig. 4 shows which BPMN concepts are supported by the core process simulator (shown in bold), whereas constructs supported as plug-ins are not emphasized. These were selected based on their usage frequency by practitioners [7]. Constructs which are currently 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. Additional to the plug-ins for advanced BPMN concepts, the simulator was extended to support batch activities [5]. A batch activity allows to bundle the execution of groups of process instances at this activity to increase process performance. The integration of bath activities

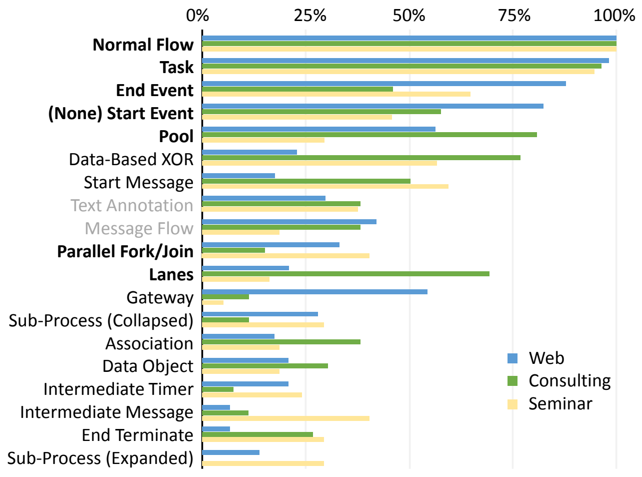


Fig. 4. List of frequently used BPMN constructs and their support in the business process simulator, cf. [7].

required 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). The developed plug-ins for supporting the batch activity are shown in Fig. 2. With the existing entry points, it was possible to adapt the execution semantics of business processes and to synchronize the execution of several process instances.

For extending the simulator, a plug-in developer can use the presented *pluggable*-classes to extend them for modifying the simulator at different stages. Therefore, a developer would generate a package to which all plug-in classes and additional needed classes are added. This package can be added as jar-file to the simulator-project. The new plug-in package, then, has to be registered in the plug-in list *\META-INF\plugins\plugins_list* which can be found in the project resources.

4 Conclusion

This paper presented an extensible BPMN process simulator with its architecture and its plug-in structure to extend and modify the basic BPMN simulator. The current simulator already includes plug-in packages for several advanced BPMN constructs which can be selected or deselected for a simulation run. Further, the entry points to the simulator were used to integrate a new BPMN concept – batch activities for batch processing in business processes. In future, we aim to extend the simulator user interface to show the generated simulator reports graphically.

A special thanks to the student Leon Bein which developed the simulator user interface.

References

1. Abel, M.: Lightning Fast Business Process Simulator. Master's thesis, Institute of Computer Science, University of Tartu (2011)
2. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A., et al.: Fundamentals of business process management, vol. 1. Springer (2013)
3. 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)
4. Knöpfel, A., Gröne, B., Tabeling, P.: Fundamental Modeling Concepts: Effective Communication of IT Systems. Wiley (2005)
5. 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)
6. Wong, T.Y.: Extensible BPMN Process Simulator. Master's thesis, Hasso Plattner Institut, University of Potsdam (2017)
7. 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)