BUSINESS PROCESS SIMULATION WITH IYOPRO AND DESMO-J

Philip Joschko, Johannes Haan, Tim Janz, Bernd Page

Department of Informatics, University of Hamburg, Germany

joschko@informatik.uni-hamburg.de, 6haan@informatik.uni-hamburg.de,
6janz@ informatik.uni-hamburg.de, page@informatik.uni-hamburg.de,

ABSTRACT
This paper describes how a BPMN model editor can be extended with simulation capabilities. In cooperation with Intellivate GmbH, which is the developer of the editor IYOPRO, our working group has developed a DESMO-J simulation library extension for simulating business processes notated in Business Process Model and Notation 2.0 (BPMN). We summarize our general approach of extending BPMN models with simulation properties and present, in more detail, problems and best practices while integrating the simulation library. Furthermore, we describe which information is needed to enhance models for simulation purposes and which experiment result can be expected.

Keywords: business process simulation, discrete event simulation, modelling tools, BPMN 2.0

1. INTRODUCTION

1.1. Motivation

The documentation, analysis and optimization of business processes are becoming increasingly important for large and medium-size enterprises (see Komus 2011). Business Process Modelling (BPM) provides a basis for communication on the processes within a company. Building upon this, Business Process Analysis enables improvement of efficiency, like reducing costs or processing time. One applicable method for optimizing processes is discrete event simulation.

For successful simulation experiments adequate modelling and well-defined system boundaries are just as necessary, as an entire data basis and knowledge about the modelled system. Additionally, for efficient modelling, a suitable tool is required that supports the modeller in his work to create models. In a best-case scenario, the business process modelling tool brings along exhaustive simulation features.

In a joint project with Intellivate Gmbh, which is a business consultancy in Hamburg, Germany, we integrated our simulation library DESMO-J ("Discrete-Event Simulation and Modelling in Java") into Intellivate's BPMN model editor IYOPRO ("Improve your Processes", www.iyopro.de). For this, we had to enhance DESMO-J with the capability of simulating business processes. This included the integration of the code into the IYOPRO editor. Moreover, we developed a model converter for generating DESMO-J models out of IYOPRO BPMN models. In addition, we enhanced the graphical user interface of IYOPRO for simulation purposes, e.g. there are user controls for experiment planning. Subsequently, the experiment results are presented as diagrams like charts and histograms, in order to provide an adequate communication base for process analysis and decision making.

1.2. BPMN

In Business Process Modelling, a graphical modelling notation is used to visualize production processes and information flows. Some of the established and popular notations for this purpose are BPMN, Event Process Chains, or Unified Modeling Language (List et al. 2006, Komus 2011). All these languages are based on a flow chart representation, where activities are mapped as nodes and the process flow is modelled by means of edges. They differ in the number of given elements and higher level modelling constructs as well as in degree of formalization.

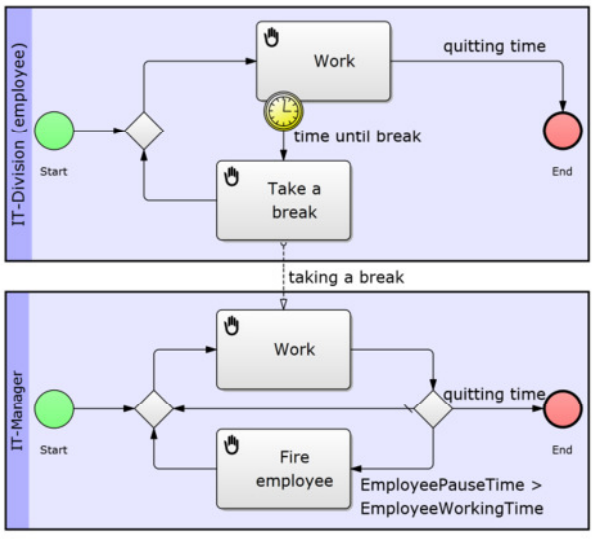

Figure 1: BPMN 2.0 example process

BPMN 2.0 was standardized by the Object Management Group in 2011 (White et al. 2011). BPMN is characterized by a particularly high number of elements (Allweyer 2009). Figure 1 shows a BPMN model with two pools, four activities, two start and two end events, an attached intermediate event, gateways, sequence flows and a message flow. Apart from different activities there are many specialized events for sending and receiving messages, signals, errors, compensations, termination and other functionalities. Especially for cancelation events, the possibility of attaching events to activities is a unique feature, which is most practical for cancelation events. Different edge types are distinguishing sequence, association and message flows. While sequence flow describes the processing order within an organizational unit, message flows describe the interaction between different processes (see Freund 2011, Wohed et al. 2006).

BPMN does not offer any simulation characteristic at all. Since in BPMN version 2.0, special attention has been paid to feasibility, in order to run models in Business process engines, it is generally possible, to transform BPMN models into simulation models.

1.3. Simulation

Simulation is a modelling methodology for investigating the dynamic runtime-behavior of a system. Mostly, the stochastic state variation in time is recorded and statistically evaluated. Simulation makes the comparison of alternative system configurations possible, without having to experiment with the actual system and thus endangering the ongoing enterprise scope.

Not all potential influences on a system can be detected nor modeled deterministically, e.g. the execution time of an operation exported by a human being is always subject to random fluctuations. This affects all variables of a system. Typical examples are order sizes, processing times or success probabilities. Nevertheless, the realistic variability of such characteristic factors can be indicated by means of stochastic distributions. However, the use of stochastics requires a sufficiently large number of simulation experiments. Thus, only the aggregation of results of several experiments will provide reliable information about a system (Page and Kreutzer 2005).

Simulating business process models provides empirically founded comparisons of alternative possible decisions, e.g. resource allocation or strategy optimization. Regardless of the chosen modelling notation, a business process model can be transformed into a simulation model, if the model is enhanced with some simulation specific properties. Particularly, stochastic properties affect the duration of activities and the interarrival time of events. The total cost of activities, the number of concurrently running processes, the duration of (sub-) processes, the length of waiting queues and the probability for the occurrence of specified events are also relevant topics. See section 4 for the entire result output of our simulation tool.

2. TOOLS

This section introduces the simulation library DESMO-J and the BPMN 2.0 model editor IYOPRO.

2.1. Simulation Library DESMO-J

Simulation software can be distinguished into two main types. On the one hand, there are integrated simulation development environments, which support the simulation study as a whole, including graphical model editors, tools for data collection, experimentation interfaces and support for statistical analysis and evaluation. These are typically commercial software tools that often focus on a specified domain. On the other hand, there are simulation libraries that concentrate on model implementation. They offer maximum flexibility to the modeler but in the same time require higher modelling skills. He/she needs substantiated knowledge in simulation technology as well as in pragramming, e.g. in object oriented programming languages. Simulation libraries can act as the core of an integrated simulation development environment.

DESMO-J (Discrete-Event Simulation and Modelling in Java) is a simulation library for developing discrete event simulation models (Banks 2010; Page and Kreutzer 2005) in the object-oriented programming language Java. DESMO-J offers a comprehensive simulation framework.

There are several ready-to-use black box classes, like a simulation clock, a scheduler or an experiment class, encapsulating the experimentation infrastructure. Further black box classes can be directly integrated into the model. E.g. there are statistic classes for collecting statistical data including counting, uniform or time-weighted aggregated samples, determining confidence intervals and generating histograms. A large set of random number distributions can be used to describe stochastic influences (see section 1.3). This set includes Normal, Continuous Uniform, Triangular, Exponential, Erlang, Discrete Uniform, Poisson and Geo Distribution. Queue classes can be used to queue entities, which are waiting for events like customer's arrivals.

Apart from the black box components, there are white box components, which have to be used to implement the system's behavior. These are abstract Java classes, which have to be supplemented to describe the system entities, i.e. there are events, processes, entities and the model container itself. DESMO-J offers mainly two simulation world views. In the process oriented world view, the modeller describes the life cycle of entities from a worm's eye perspective. Within this life cycle a process can be render existing processes passive or reactivate them. In contrast, in the event oriented world view, he/she describes the effect of events from a bird's eye perspective. This can mean the manipulation of global model states or scheduling of further events. DESMO-J does not enforce a decision for either event or process modelling. The user is free to

combine both modelling styles. Beyond that DESMO-J can also deal with transactions as well as activities.

The expandability is an important aspect in using DESMO-J. It is possible to extend it by additional class libraries and the integration into software frameworks or applications. Domain-specific extensions, which provide objects for the particular domain, are developed for DESMO-J to ease modelling and simulation in these special application domains.

Several commercial business process modelling tools use DESMO-J as simulation engine, in order to support such analysis. Unfortunately, we cannot provide a complete list of software products including DESMO-J, since institutions do not necessarily get in touch with us if they use DESMO-J. Mostly, we find out by coincidence that DESMO-J has been integrated into a simulation suite. From our knowledge, DESMO-J is a part of Tibco Business Studio, Borland Together, eClarus Business Process Modeler for SOA Architects and recently IYOPRO, our favorite in user friendliness.

2.2. IYOPRO

IYOPRO (Improve Your Process) is a BPM Suite for modelling business processes, managing tasks and human workflows. It supports the design and the execution of business processes in a single application with an ergonomic user interface. The BPMN 2.0 standard is used as modelling language for this purposes and completely covered by IYOPRO. A validation engine gives feedback on modelling errors, as non-standard-conform combination of elements or missing modelling information. IYOPRO includes a business process execution engine, which enables the integrated execution of well-defined processes. These processes can interact with several tools, like user-forms for entering data, or automated data import from external systems. Process models are grouped with the help of a solution explorer. IOYPRO can handle collaboration diagrams, choreography diagrams, conversation diagrams (these three diagram types are part of the BPMN 2.0), process maps, organization diagrams, user form definitions, and database schema definitions. Solutions and projects can be stored in an online repository, which supports collaboration capabilities as version controlling and team access rules. Alternatively, projects can be saved as XML-files on local hard drives. As Silverlight application, IYOPRO is usable in browsers but also as stand-alone application. It is also available as Software as a Service application. There is a free base version available online and professional licenses are provided for advanced purposes.

3. EXTENDING BPMN 2.0 FOR SIMULATION EXPERIMENTS

In this section we describe in detail what additional information is required for a simulation of BPMN models. None of these properties are part of BPMN 2.0 specification.

3.1. Duration of Activities

The most crucial components regarding the lead time of business processes are the activities. Due to its nature, an activity consumes time while being executed. We call this the duration of an activity. This circumstance is represented by applying a DESMO-J stochastic distribution to an activity. Considering his needs the user can parameterize one of the well-known distributions to determine the duration of every activity. Furthermore, it is also possible to apply definite points in simulation time. This can be useful when modelling a work schedule for human performers in an office.

3.2. Interarrival Time of Events

Similar to activities events have a high impact on the lead time of business processes, due to waiting times between receiving and sending events in particular.

Every time a start event is triggered a new process instance will be created. Some start events are triggered by the receiving of messages, signals, etc. They do not need any further information to be set, because their occurrence just depends on the corresponding sending events and the interarrival time can be calculated automatically. In contrast, general and timer start events do not have corresponding sending events. Their interarrival time results from stochastic distributions that have to be set by the modeler just like the duration of an activity described above. E.g. you have a process that describes customer's behavior in your online shop. You can then describe customer's arrival with the help of a normal distribution (or any other of DESMO-J's distributions). You need at least one general or timer start event in your simulation models. Most of the processes that require general or timer start events involve the receipt of an order.

3.3. Evaluation of Process Properties

Decisions within a process depend on conditions that result from the input into the process, from global states or from calculations during process run time. Additionally, in simulation experiments, process properties can depend on stochastic influences for mapping non-deterministic influences. Examples for such kind of additional information are order sizes, priority ratings of customer's inquiries or calculated costs of optional activities. Which path is followed at exclusive and inclusive gateways mostly depends on such properties.

Properties are represented by process variables, which are composed of a name and a value. Only the process instance itself has access to its variables, unless the instance transported these values explicitly to the outside.

These variables can be assigned or manipulated by activities or by data objects attached to events. New values can be recomputed by means of Python scripts, which offers complex mathematical calculations and program instructions like loops and if-conditions.

If a process instance needs information from another, e.g. the warehousing needs information from

the sales section, local variables can be transferred in messages between pools / process instances. They also form the data basis for decision-making at conditional (inclusive and exclusive) gateways. This is done by evaluating the expressions on the outbound links of the gateways (e. g. link_1: x==1; link_2: x>1).

Regarding the BPMN model in Figure 1 we have got an exclusive gateway, which makes its decision based on the simulation time (quitting time or not) and on the accumulated time of pauses for a specific employee.

We must pay attention here though: The IT manager from the bottom pool can finish his or her work activity either by accomplishing the working time for the day or by receiving a message from the employee / upper pool.

Based on the nature of coincidence however, it is possible that the IT manager quits working for the day and at the same (discrete simulation) time receives a message from an employee and the employee's pause time is lower than his working time. If this happens, there will be a DESMO-J error at run time. DESMO-J will terminate this specific process instance and it will add an entry in the error log regarding this particular matter. (See section 4 for more information on the Error Log.) The modeler will have to find a way to get rid of this flaw, because his simulation result will not be valid.

3.4. Assignment of Resources

Resources are used to map real life processing entities in models, especially simulation models. Activities usually need a resource like an employee or a machine to perform a certain task.

Assigning a limited resource to an activity exposes capacity restrictions (bottlenecks) and can be a critical indicator to the lead time of a process. Bottlenecks occur when a resource is working at full capacity. Process instances that arrive at the activity would have to wait until the accumulated number of free resource entities is in accordance with the required value. On the other hand there can be under-utilized resources, whose number could be decreased to reduce costs.

3.4.1. Modelling and Mapping of Resources

Figure 2 illustrates the human resources of an enterprise as organigram. In this case it is the organizational structure (including the persons' roles) of the project described in this paper. Mr. Page is the boss, Mr. Joschko manages the project and Mr. Janz and Mr. Haan are the two programmers responsible for the design and the implementation of the desired features.

The roles and the number of entities are actually the most important part in this representation, for they will be mapped to resources before the simulation starts. Each resource contains a list of attributes to specify its characteristics. In our example there are only two significant properties of a resource: Human and Role. Mr. Janz and Mr. Haan are two identical resources from the point of view of the simulation. Mr. Page and Mr. Joschko on the other hand cannot be compared to each other nor to the programmers.

After their creation the new resources must be assigned to a resource pool, which will be responsible for the provision of resources throughout the simulation. It has to be considered however that some resources are only available in a particular scope. This means a resource is only available at a certain activity, swimlane or pool. Resources cannot be used beyond the boundaries of their scopes.

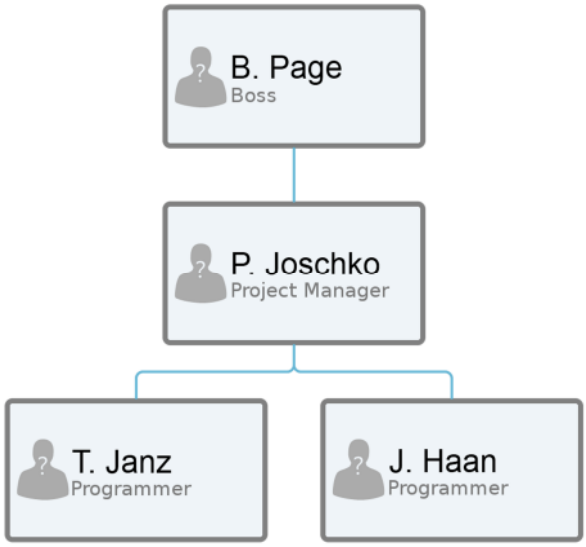

Figure 2: Organigram describing the Human Resources of an Enterprise

3.4.2. Execution Example

After creating such an organigram, you can add its resources (in this case: roles) to the tasks of your BPMN model. In the following we will describe how this can be done for the model depicted in Figure 1.

We assign one programmer role to each task from the upper pool (IT Division). This means that the task can only be started, when a resource of the role "programmer" is available. Every process instance reaching a task that needs a resource, checks whether the resource pool can deliver it. If the desired resource is available, the process instance will take it and lock it for the time of usage, so that it cannot be utilized by another process instance. If the resource pool is not capable of providing the designated resource, the demanding process instance will be inserted into a waiting queue until its request is met.

The tasks from the bottom pool are assigned one project manager each. Additionally the "fire employee" task has to be allocated with one programmer resource, which will be unlocked by force, if it is being locked by another process instance.

4. SIMULATION REPORTS

After running a simulation experiment the simulation report is generated. This report consists of several key performance indicators (KPI) like:

- (Total) lead times for business processes and activities
- Information to the decision-making at conditional gateways
- Capacity utilization for resources
- Interarrival times of events
- Maximum, minimum and average values for the DESMO-J queues and stochastic distributions

Since sometimes it is more obvious to identify facts by looking at a compact graphical representation than at some detailed numerical KPIs, we implemented a variety of charts:

- Pie charts for decision-making at conditional gateways
- Bar charts for total lead time of processes (Figure 3)
- Boxplots (box-and-whisker diagrams) for lead time of processes and activities (Figure 4)
- Pie charts visualizing the time, resources are idle, in use or waiting (Figure 5)

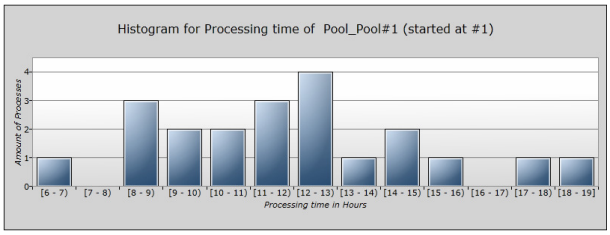

Figure 3: Bar chart for the lead time of a process

Which KPIs and charts are important for a conclusion depends on the particular motivation. Thus many different statistics are offered for answering ones questions. Some of them are expensive in memory usage or computationally intensive. These are not created automatically but can be switched on a particular simulation experiment.

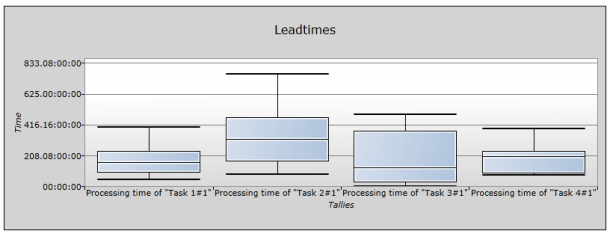

Figure 4: Boxplot for the lead times of activities

Besides the simulation report, an error log and a trace are generated. The trace contains the step-by-step process of the simulation run, e.g. which activity was started at which point of simulation time. It is useful to comprehend how statistics in the report result from certain events.

The generated reports are only valuable and applicable, if the simulation model was designed validly and with the appropriate level of detail. Therefore we cannot stress enough, how important it is to construct a model that adequately represents the real life situation.

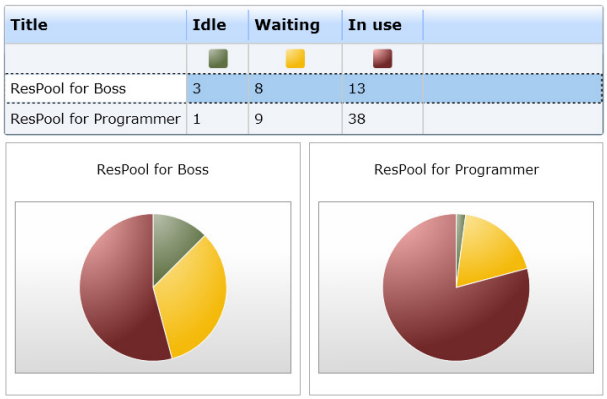

Figure 5: Idle, Waiting and In Use times of resources

5. IMPLEMENTATION

In this section we will describe the software technical aspects of integrating DESMO-J into IYOPRO. Furthermore this chapter will cover the problems encountered during this process.

5.1. C# Port of DESMO-J

Since IYOPRO is a Silverlight web application, we ported our Java-based DESMO-J into C# programming language. C# language is very similar to Java - both languages use the C++-Syntax. The main task is to map classes (e.g. containers like lists) from the Java Standard Library to classes from the .NET framework. Problems arise because some programming concepts in Java work differently than in C# (e.g. the concept of generic classes). Since DESMO-J is constantly improved and enhanced, maintaining two branches of DESMO, one in Java and on in C#, would be too costly to be feasible. Nevertheless, having a .NET version of DESMO-J which is always aligned with the maturity level of the Java version is desirable. We implemented an Ant-script that automatically ports the high level language Java into high level language C#. Of course, this script is adapted to our needs, and not in a position to transform arbitrary code. As a result, we are now able to generate C# source code that is nearly equivalent to our Java-based DESMO-J.

5.2. BPMN Extension – General Approach

We took advantage of DESMO-J's extensibility, to implement a BPMN extension incorporating the semantics of BPMN elements in order to make them executable. Each individual BPMN element is derived from appropriate DESMO-J white box classes. E.g. there is a special BPMN-process derived from DESMO-J's SimProcess class while most of BPMN flow elements, like activities, several event types and sequence flows are derived from DESMO-J's Entity class. Furthermore, the library includes message flows,

pools, swimlanes and data-objects, which are also derived from DESMO-J's Entity class.

The BPMNProcess class represents process instances. A new process instance will be created each time, when a start event is triggered. Within the life cycle, there is a loop, which works through the elements of the sequence flow. Every sequence element has a method "Execute", which is called by the BPMNProcess instance. The Execute-method of the BPMNActivity class takes a value from the stochastic distribution for the duration and renders the process passive for the given time. The Execute-method of receiving BPMNEvents checks whether the event has already been triggered. Otherwise, the process is queued (depending on the type of the event), and rendered passive until the event occurs. The Execute-method of sending BPMNEvents checks whether there are processes waiting for the specified events. If so, the corresponding processes are then reactivated and their sequence flow will be continued.

When a simulation experiment is started, the graphical model is converted into a simulation model and for each BPMN element (pools, edges, nodes) the corresponding DESMO-J object will be instantiated. The created element structure cannot be changed at runtime. All process instances work on the same BPMN object instances. Only dynamic objects such as process instances or messages are created at simulation run-time. When modelling errors are detected at this point (e.g. missing start events), the simulation run is stopped and the user gets a corresponding notification.

5.3. Gateways

Gateways are used to split or join sequence flows. In parallel splits each outgoing path is followed. For this, new child processes are created that execute concurrently. Since the parent process is associated, it terminates only if every child process has terminated. The implementation of parallel, exclusive and inclusive splits was very easy with this approach. Also the parallel and exclusive join is very simple. The incoming processes are terminated and a new child process is created. This happens when each child process has arrived at the gateway (parallel join) or each time when a child process reaches the gateways (exclusive join).

The implementation of the inclusive join was complex. This gateway fires when at least one child process has reached the gateway, and no further child processes exist that may arrive at the gateway. This means that the gateway does not fire only when a process reaches the gateway, but also if a child process terminates or chooses a sequence path, which makes it impossible to reach this gateway. Therefore, an inclusive join has to sign up for each child process, which is able to reach that inclusive join. The child process will then notify the gateway, if reaching of this gateway becomes excluded. The modeller has to note, that using inclusive joins is extremely susceptible to deadlocks when it is applied within a sequence loop.

5.4. Message flows

BPMN 2.0 distinguishes between sequence and message flows. While sequence flows only exist within a pool, message flows connect activities or events from different pools. This is used to model the interaction between process instances. Since multiple process instances of a pool can exist at the same time, it must be clarified, for whom the message is destined.

If there is a pool A and corresponding process instances A1 and A2, and a pool B with corresponding process instances B1 and B2, every of this process instances references an address book, which includes a reference to one process instance for each pool. When process A1 was started, its address book only holds one reference: the process instance itself (A1) and the corresponding pool A. There is no reference for a process instance from pool B at this point. If a message is sent to pool B, the first process instance (B1) arriving at the receiving event, not having any entry in its address book for pool A, may take the message. The address books of A1 and B1 are then merged and thus the address book of A1 holds a reference of B1 and vice versa. If another message is transferred from A1 to pool B, only the process B1 may take the message. If B1 sends a message to pool A, only A1 may take the message. There is no possibility to send a message from A1 to B2 anymore.

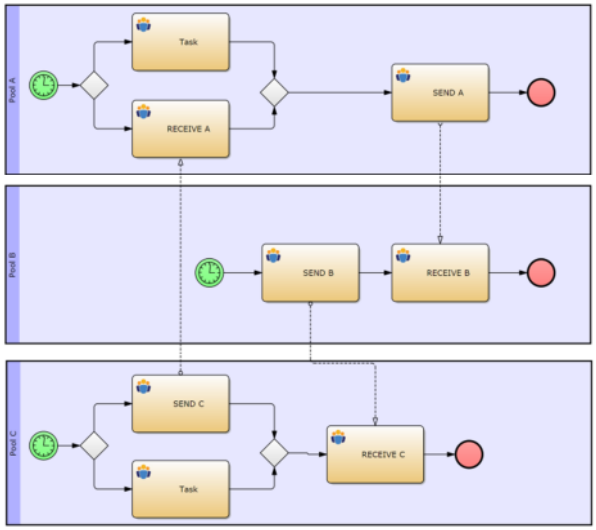

Figure 6: Potential address book conflict

This simple set of rules is not sufficient however. If there are three pools A, B and C (see Figure 6), and there is an optional message flow between pool A and C, a message flow between A and B, and a message flow between B and C, address book conflicts will occur: The address book of A1 has an entry for pool C (C1) and no entry for pool B, the address book of B1 also has an entry for pool C (C2). If process B1 receives a message from A1, there is a reference conflict, because there are two references for pool C (C1 and C2). That's why we implemented a rule, eliminating this behavior: A process instance may only take a message if a) the sender is listed in the receivers address

book, or b) the receivers address book does not contain any pools which are listed in the sender's address book.

5.5. Attached events

Another feature of BPMN 2.0 is the ability to attach receiving events to activities. They exist to describe, what happens if an event occurs while an activity is executed. There are interrupting events, which abort the execution of the activities, and non-interrupting events. In our implementation, for every attached event a child process is started. The child process announces itself to the corresponding sending events and waits for their occurrence. If the activity has finished before the event occurs, all child processes are deleted. If the event occurs before the activity has finished and it is marked as interrupting, the child process, which executes the activity, is destroyed and the child process, which is connected to the attached event, is executed. This works for all intermediate events that we have implemented, i.e.: message events, signal events, timer events, error events, cancel events and escalation events. Just the compensation event has a deviating behavior. It can also be executed when the corresponding activity is not alive anymore.

5.6. Integration into GUI

DESMO-J BPMN extension was designed as a programming library that was integrated into the graphical model editor IYOPRO. We had to make this functionality accessible on the user interface. To run a simulation experiment, some experiment parameters have to be set. First, you need a stop time, which describes the length of a simulation run, e.g. you can simulate 24 hours or a complete business year. Additionally, you need to set a seed for stochastic distributions. Although, you can set a separate seed for every distribution, you will just set an experiment seed, which will be used to calculate a seed for each distribution. If an experiment is repeated with the same seed, the same random numbers are drawn.

While the BPMN model editor component was reused without any changes, we had to enhance the property editor with the element properties described in section 4. Also the resource editor and the Python script editor were used without any changes. All logical DESMO-J BPMN elements contain a reference to their corresponding graphical objects. Therefore, the experiment results can be linked to the corresponding graphical objects. If the modeller clicks on a result table, the graphical object is highlighted in the editor. If a simulation error occurs (e.g. there are no receiving events for a sending event), the according graphical element is also highlighted, so that the modeler can pinpoint and fix this error easily.

6. DISCUSSION

We implemented a BPMN extension for our simulation engine DESMO-J, which incorporates the semantics of the most common BPMN elements, like events, activities, pools, swimlanes, data objects, sequence and message flows. This allows the calculation of indicators for process lead times, waiting and interarrival times. When a resource model is connected to the business processes, the utilization of resources and waiting times for resources can also be calculated.

Although the simulation functionality already works and can be used for simulating business processes, our work is not completed yet. There are still some BPMN elements missing in our DESMO-J BPMN extension, as there are conditional events, event-based or complex gateways.

Unfortunately, the simulation properties are not part of the BPMN standard. While BPMN models can be manipulated, exported and imported with different editors, all simulation properties get lost, if you open the BPMN models in another editor. This will not change in the near future.

We plan to implement assistant concepts that can help the user in generating simulation models and in defining experiment plans. Additionally, there will be some visualization features for investigating simulation experiments at runtime. We are also working on instruction material that documents the simulation functionality properly and thus supporting the modeller.

Since the BPMN extension is a class library, we can reuse it in other software products. E.g. we are currently working on a simulation suite for the operation stage of offshore wind parks (www.systop-wind.de) coupling business process models with domain specific models of offshore wind parks, like weather or a maintenance models.

The current status of our work on IYOPRO can be tested at www.iyopro.de. If you are interested in the DESMO-J BPMN extension feel free to contact us or Intellivate GmbH.

ACKNOWLEDGMENTS

We would like to thank Intellivate GmbH for the constructive and inspiring cooperation.

REFERENCES

Allweyer, T., 2009. *BPMN 2.0 – Business Process Model and Notation*. Norderstedt, Germany: Books on Demand GmbH.

Banks, J. et al. 2010. *Discrete-Event System Simulation*. Upper Saddle River, New Jersey: Prentice Hall.

Freund, J. and B. Ruecker, 2010. *Praxishandbuch - BPMN 2.0*. 2nd ed. Munich, Germany: Carl Hanser Verlag.

Komus, A., 2011. *BPM - Best Practice*. Berlin, Germany: Springer.

List, B., Korherr, B.. 2006. An Evaluation of Conceptual Business Process Modeling Languages. *Proceedings of the 21st ACM Symposium on Applied Computing (SAC'06),* pp. 1532-1539. April 23-27, Dijon (France).

Page, B., and W. Kreutzer. 2005. *The Java Simulation Handbook – Simulating Discrete Event Systems with UML and Java*. Aachen, Germany: Shaker.

White, S. et. al., 2011. *Business Process Model and Notation*. Object Management Group, Available from: http://www.omg.org/spec/BPMN/2.0/ [accessed 15 May 2012]

Wohed, P., van der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M. and Russell, N. 2006. On the Usability of BPMN for Business Process Modelling. Lecture Notes in Computer Science 4102, 161-176.

AUTHORS BIOGRAPHY

BERND PAGE holds degrees in Applied Computer Science from Technical University of Berlin, Germany, and from Stanford University, USA. As professor for Modeling & Simulation at University of Hamburg he researches and teaches in Discrete Event Simulation and Environmental Informatics. He is the head of the working group that developed the simulator DESMO-J and author of several simulation books.

PHILIP JOSCHKO studied Computer Science at the University of Hamburg. He works as a scientific assistant and PhD candidate in the Modelling & Simulation workingroup of Prof. Dr. Page. His research interests are business process simulation, simulation software development, and the application domain of offshore wind parks. Since 2005 he takes part in improving DESMO-J. He applied DESMO-J in several simulation projects.

JOHANNES HAAN studies Information Systems at University of Hamburg. After receiving a Bachelor of Science degree in 2010 he is currently engaged in his master thesis. He is working in the project presented above on a part-time basis.

TIM JANZ studies Computer Science at University of Hamburg where he received a Bachelor of Science degree in 2010. He worked in the presented project with simulation using DESMO-J for his bachelor thesis and additional projects. Alongside his studies he is working in this project on a part-time basis.