A Tool for Animating BPMN Token Flow

Thomas Allweyer and Stefan Schweitzer

Fachhochschule Kaiserslautern, Fachbereich Informatik und Mikrosystemtechnik,
Amerikastr. 1, 66482 Zweibrücken, Germany
thomas.allweyer@fh-kl.de, stsc0048@stud.fh-kl.de

Abstract. The concept of tokens flowing through a process model is very useful for explaining and understanding the meaning and the execution semantics of a BPMN model. This paper presents a software tool for animating the token flow of arbitrary process models. It can handle different scenarios of gateway combinations, loops, expanded and attached sub-processes, untyped start and end events, as well as terminating end events. It is possible to show several process instances within the same model. They are represented as differently colored tokens.

Keywords: Animation, BPMN, E-Learning, Execution Semantics, Sequence Flow, Token Flow.

1 Introduction

The semantics of BPMN sequence flow can be explained by the concept of tokens flowing through the model. The BPMN specification introduces tokens as a theoretical concept "as an aid to define the behavior of a process" [1]. The instantiation of a process is represented by the start event producing a token. This token then travels through the sequence flow until it is consumed by an end event. When it reaches an exclusive split, the conditions of each alternative flow are evaluated, and the token is routed to exactly one of these alternative flows. When a token arrives at a splitting parallel gateway, it is duplicated, and the outgoing sequence flows receive one token each. A joining parallel gateway will emit one token after all parallel tokens have arrived; i. e. after each entry has received a token. Possible misinterpretations of the meaning of parallel flows, such as the need for simultaneous processing or simultaneous completion of parallel activities, can be corrected very easily by explaining the behavior of the tokens.

Other BPMN elements, such as inclusive gateways or sub-processes, can also be defined very precisely with the token flow concept. This concept has also proved to be very useful for teaching and learning BPMN. Several BPMN books use token flows for explaining the meaning of process models, e.g. [2], [3], [4], [5].

The token flow concept represents the execution semantics of BPMN models, i. e. the token flow describes how a process engine would execute the model. Although many models are not executable, every BPMN model should be not only syntactically correct, but also semantically. The sequence flow should correctly

J. Mendling and M. Weidlich (Eds.): BPMN 2012, LNBIP 125, pp. 98–106, 2012.
© Springer-Verlag Berlin Heidelberg 2012

reflect the modeler's intentions of how the process is actually performed. Ambiguities, different interpretations etc. can be resolved, if the token flow is analyzed – and errors can be detected, such as wrong gateway types.

As the token flow is a theoretical concept, its analysis is usually only done mentally. The analyst just imagines how tokens are created, routed, duplicated, merged and consumed. Such analyses are neither carried out systematically, nor regularly. For executable processes, modeling errors may be detected during testing the implemented process. However, testing does not detect all errors, and it would be cheaper if modeling errors could be detected earlier.

For non-executable processes, such modeling errors may not be detected at all. There are modeling tools which help detecting syntactical modeling errors. It is also possible to automate the analysis of the execution behavior in order to find problems, such as dead locks [6, p. 267 ff]. Modeling errors that are much harder to detect are deviations of the model's actual behavior from the intended behavior. Systematic token flow analyses could help the modeler recognizing whether the model behaves as expected.

Such a systematic token flow analysis can be supported by visually animating the token flow. Especially for beginners it is difficult to mentally "play through" a model. Therefore, many trainers use animated presentation slides in BPMN courses for visualizing the token flow of simple models, and there are also some web sites containing process model animations, e.g. [7], [8].

In this paper we present a lightweight tool which can be used for animating arbitrary BPMN models. It covers the most important basic sequence flow elements, and it can handle models of varying complexity. The tool can be used for introductory BPMN courses, for self-learning and experimenting with different process designs, but also for analyzing process models in industrial practice.

The remainder of the paper is structured as follows: In chapter 2, we discuss related work. The requirements for the tool are discussed in chapter 3. Chapter 4 describes the developed animation tool. Chapter 5 concludes the paper and gives an outlook on further work.

2 Related Work

The execution semantics of a process model can be analyzed by actually executing the model in a process engine. However, process engines are rather complex and often expensive, and for making a process executable, a lot of additional artifacts need to be developed, such as data structures and variables, rules and conditions, user interfaces, service calls etc. [9], [10]. This is only an option for analyzing models which are to be executed anyway, but not for BPMN models that are not meant to be executed. Besides that, not all process engines vendors have implemented the BPMN standard completely. In many cases the execution semantics deviates from the standard. Some BPMS vendors have included animation features into their systems, e.g. Inubit [11] and IYOPRO [12].

Another means to analyze process models is provided by simulation tools. The objective of a process simulation is to analyze the dynamic behavior in respect to load distribution, resource capacities, waiting queue lengths, and cycle times [13], [14],

[15]. This means that the entire system with many process instances is considered, but not the process logic of a single instance. For example, at an exclusive split it is not important which conditions are true, but only the percentage of instances in which a specific path is selected. Therefore it is not possible to analyze the actual flow logic of single instances. Some simulation tools also provide animation components, e. g. iGrafx Process [16] and L-SIM [17]. However, a simulation requires a lot of information, such as statistical distributions of process instantiations, probabilities for selecting specific paths, etc. All this would not be required for analyzing the execution semantics of a model.

We have already mentioned that there are various tools containing syntax and rules checks, e. g. there is a tool for checking the modeling rules defined by Bruce Silver [18]. There are also several tools for business process verification, e. g. [19], [20], [21]. Such tools may help discovering some of the problems that can also be detected by analyzing the token flow, but the modeler neither can see the token flow in the incorrect model, nor in the corrected model, and it cannot be detected that an otherwise correct process model simply describes something else than intended.

Recently, some BPMN modeling tool vendors have included animation features in their tools. An example is "Innovator for Business Analysts" from MID [22]. However, most of these animation features have some drawbacks. "Innovator for Business Analysts", for example, does not duplicate tokens at a splitting parallel gateway, but it lets the user decide which *one* of the parallel paths he wants to animate. Such an animation component is more a presentation and discussion aid rather than a correct token flow animation. The modeling component of IYOPRO provides a better implementation of the BPMN semantics, but the animation is restricted to one process instance at a time. In all commercial systems, the animation feature is tightly integrated into the system, and it is not possible to access or change the implementation of the animation logics. There are also several research prototypes that include some kind of BPMN token flow animation of BPMN, e.g. [23], [24]. However, these prototypes are usually focused on analyzing specific questions concerning the model semantics rather than providing a general-purpose animation component.

3 Requirements

The main purpose of the tool is to visually animate the token flow of arbitrary BPMN models for presenting and analyzing the model's execution semantics. The tool is aimed at BPMN trainers, learners, and active modelers. BPMN trainers should be able to prepare demonstration models for their courses. Such models can be used for explaining the various BPMN elements. Trainers can also ask questions concerning the behavior of a specific model and use the animation afterwards in order to validate or correct the assumptions of the course participants.

The tool can also be used for self-learning. The learners can create their own models and test their assumptions by animating and experimenting with different process designs. In this way, the tool supports a more interactive and therefore more efficient way of learning than by just reading a book.

Active BPMN modelers can use the tool for analyzing their models and ensuring that the execution semantics correctly represent the intended behavior of the process.

In order to fulfill these purposes for a broad target group, the following requirements have been identified:

- The tool should be lightweight and independent of a specific platform and of a specific modeling tool. Although the seamless integration into a modeling tool would provide more comfort to the modeler, it was decided to develop an independent animation tool, so that it is useful for many BPMN practitioners regardless of their favorite modeling tool.
- It should support the BPMN 2.0 standard, using the standard exchange format as input. The models will be created with an existing modeling tool, exported in the BPMN 2.0 XML format, and then imported into the animation tool, where it will be displayed and animated. It will not be possible to modify the model in the animation tool.
- The tool should be able to visualize the token flow of various BPMN models.
- It should support the most important sequence flow elements, namely untyped start and end events, terminating end events, tasks, sub-processes (collapsed and expanded), gateways (data-based exclusive, parallel, inclusive), conditional flows and default flows (both originating from gateways and from activities).
  This selection neither is exactly identical to the common core set as identified in [25], nor to the "descriptive process modeling subclass" as defined in the BPMN specification [1]. Since some aspects in these sets are rather different to handle by an animation tool (e. g. handling correlation information for message flows), we have focused on pure sequence flow-related elements, also including some elements from the analytical subclass of BPMN 2.0.
- Important BPMN elements that do not directly affect the token flow should also be displayed, namely pools, lanes, groups, and annotations.
- The tool should ensure correct token flow according to the BPMN specification.
- It should be possible to animate the tokens of multiple instances of the same process.
- It should be possible to interactively toggle the conditions of conditional flows (at gateways and activities) between *true* and *false*.
- Since the tool's purpose is the graphical animation, it is not required to support very large models, but only models of a size that fit onto a typical computer screen (with the element labels still readable).

4 BPMN Animation Tool

The BPMN animation tool has been implemented in Java. It is a standalone tool with a Java Swing user interface (Fig. 1).

For importing model files, the JAXP DOM interface is used. The model file is validated against the BPMN 2.0 XML schema. Unfortunately, the BPMN 2.0 standard exchange format is not used consistently by different BPMN tools. Therefore it was decided to use one modeling tool as the reference implementation. We selected the Signavio Process Editor [26] as reference tool, because this tool has a rather good implementation of the BPMN exchange format, and there is a free academic version available for research and teaching. This restriction to a reference tool means that we could not entirely meet the requirement of tool independence. However, this is due to the insufficient implementation of the BPMN standard by modeling tool vendors. By

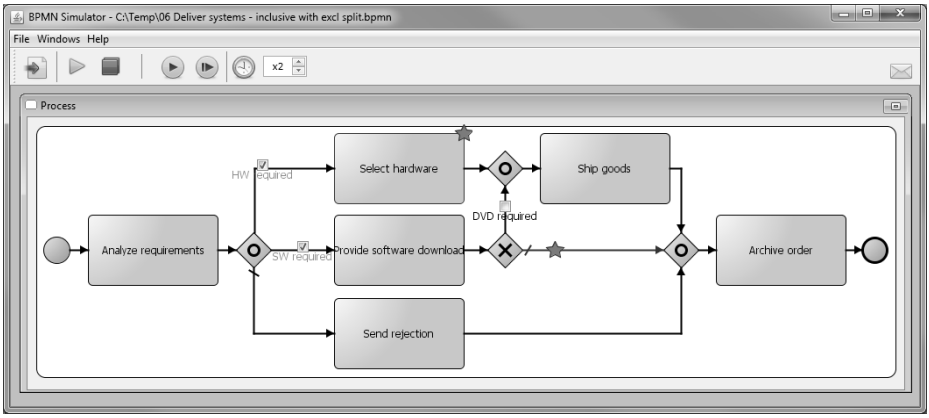

Fig. 1. User interface of the BPMN animation tool

using the standard exchange format, the animation tool is prepared to use the BPMN export from any tool that has implemented the standard correctly.

The internal structure is a rather straightforward implementation of those parts of the BPMN 2.0 meta-model that are relevant for the animation. The animation is handled by those BPMN elements that can contain or transport tokens. Each of these elements holds a list of its current tokens. At discrete time intervals it forwards tokens according to the BPMN execution semantics. The applicable execution rules are implemented in each element class.

Although the tool does not perform a complete syntax check on imported models, it checks for typical problems that prevent the model from being animated. For example, sometimes in the modeling tool a sequence flow connector is not really attached to the target object.

As Fig. 1 shows, the tokens are represented by little stars that move along the sequence flows. A process is instantiated by clicking on a start event or via the start button. When a task is activated, the star is shown on the task's upper right corner. A process can be instantiated multiple times. The tokens of different instances can be distinguished by different colors. Tokens of the same color are duplicated for parallel paths. Likewise, at parallel or inclusive joins, a token can only be joined with other tokens of the same color. At exclusive and inclusive splits there are tick boxes at the conditional sequence flows, so that it is possible to interactively change which conditions are *true*.

The animation can be paused and resumed, and the animation speed of the token flow can be changed. It is also possible to use a single step modus. The stop button removes all tokens from the model.

If one of several tokens with the same color reaches an end event, it is removed from the model. The number of removed tokens is shown in the upper right corner of the model, until the last token reaches an end event. When a token reaches a terminate end event, all tokens are immediately removed. At parallel or inclusive joins the waiting tokens of each color are displayed at the gateway. When all required tokens of the same color have arrived, they are removed and one resulting token is emitted at the outgoing sequence flow.

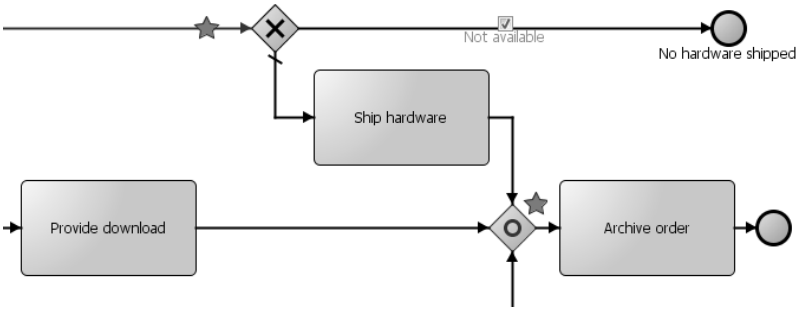

Fig. 2. The inclusive join waits for two tokens

The inclusive join has been implemented as defined in the BPMN specification. Thus, a situation as shown in Fig. 2 will be handled correctly. Here, the joining inclusive gateway waits for two tokens. One token has already arrived at the gateway.

When the other token moves to an exit of the exclusive gateway that leads to an end event, this token cannot reach the inclusive gateway any more. Therefore, the inclusive gateway does not wait any longer and immediately removes the single waiting token and forwards one token (Fig. 3).

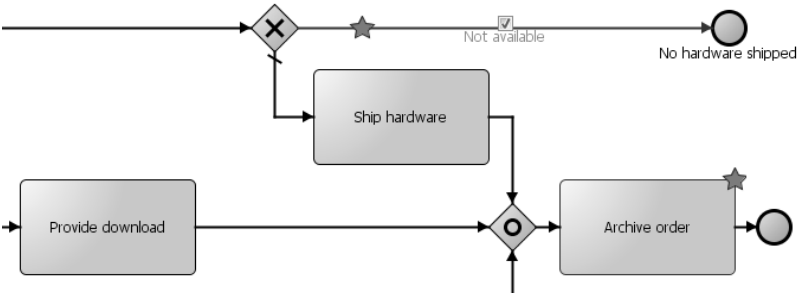

Fig. 3. The token at the top has left the path to the inclusive join, therefore the inclusive gateway has emitted a token. This token has activated the task "Archive order".

The tool can also handle the incorrect case that multiple conditions at an exclusive split are *true*. This can happen if the condition statements are defined in a way that a process variable may hold values for which more than one condition statement evaluates to *true*. For example, if the two condition statements are "*amount* > *€ 5.000*" and "*amount* ≤ *€ 6.000*", both conditions are true for amounts between € 5.000 and € 6.000. Obviously, such a case is a mistake by the process designers. In the animation tool this – usually undesired – situation can be created by marking multiple exits of an exclusive gateway as *true* (Fig. 4).

According to the BPMN specification, the conditions at an exclusive gateway are evaluated one after the other. The first one that evaluates to *true*, determines which path will be taken. Since there is no prescribed order of the exits of a gateway, the selected path depends on the internal representation of these exits in a process engine – or in this case in the animation tool. This means that exactly one path with a *true* condition will be taken, but it is undefined which one.

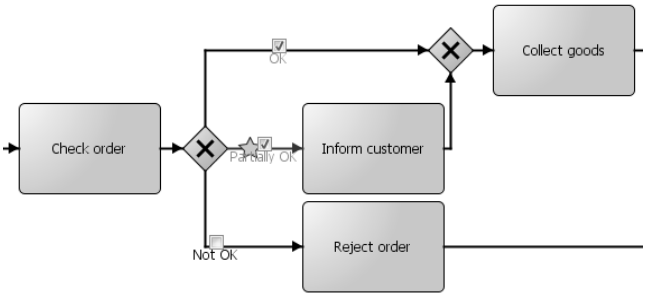

Fig. 4. The conditions of two alternative paths have been incorrectly marked as *true*. The tool routes the token to the first path it finds that is marked *true*.

If, on the other hand, no condition evaluates to *true* (and there is no default exit), the BPMN specification states that an exception will be thrown. In the animation tool, this incorrect situation is marked by a warning sign (Fig. 5). The process animation does not include explicit exception handling, but the user can handle this situation by either terminating the entire animation, or resetting the gateway exception state by marking one of the exits as *true*.

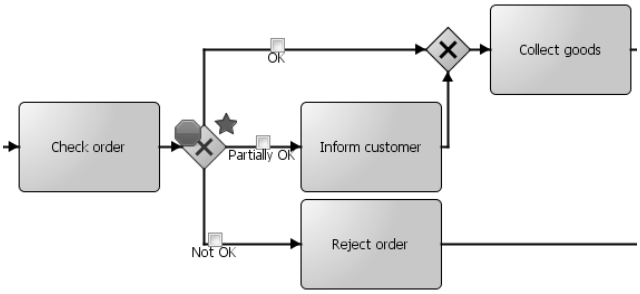

Fig. 5. The tool displays a warning sign, because no condition is marked as *true* and the process cannot continue

The tool can also animate the token flow in sub-processes, shown either in an expanded or collapsed view (Fig. 6). It is also possible to animate the token flow in a multi-level hierarchy of sub-processes.

For collapsed sub-processes, the detailed flow is shown in a separate window. While the sub-process is active, the token of the parent process is shown in the upper right corner of the collapsed sub-process. In the sub-process of Fig. 6, the parallel gateway emits two tokens. The token in the parent process does not continue its travel before both tokens have been consumed by the end events of the sub-process.

The tool has been released as Open Source under the Apache 2.0 license. It can be downloaded at http://code.google.com/p/bpmn-simulator. Instructions, sample processes, and videos can be found at http://www.kurze-prozesse.de/en/download-bpmn-token-flow-animation-tool.

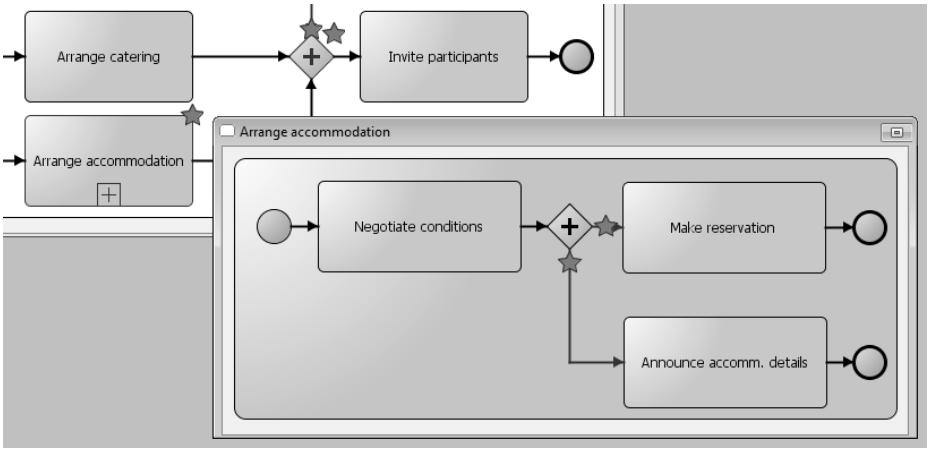

Fig. 6. Token flow in a collapsed sub-process. The detailed flow within the sub-process is shown in a separate window.

5 Conclusions and Outlook

The tool has been implemented successfully. A series of tests has shown that it can handle even rather complex models correctly. The objectives and requirements as stated in chapters 1 and 3 have been met. The only drawback was the insufficient implementation of the BPMN 2.0 exchange format by some tool vendors. Therefore we had to restrict the supported exchange format to the export format of one reference tool. However, since we use the standard format, it will not be difficult to extend the number of supported tools, as far as they fully support the standard.

Possible future extensions may include extended interaction features and further BPMN elements. For example, it could be useful to change the processing times of tasks, so that it is possible to change the order in which parallel tokens arrive at a gateway. Further supported BPMN elements could include intermediate events, attached events, and message flows.

The tool has been used successfully in several introductory BPMN courses. It could be useful to evaluate the benefits of the tool in different learning scenarios. For example, does it actually improve the learner's understanding of BPMN models, and does it help reducing the number of errors made by novice modelers?

References

1. OMG (ed.): Business Process Model and Notation (BPMN) Version 2.0. OMG document number: formal/2011-01-03 (2011), `http://www.omg.org/spec/BPMN/2.0/PDF`
2. Allweyer, T.: BPMN 2.0. Introduction to the Standard for Business Process Modeling. BoD, Norderstedt (2010)
3. Briol, P.: BPMN 2.0 Distilled. Lulu, Raleigh (2010)
4. Freund, J., Rücker, B.: Praxishandbuch BPMN 2.0, 3rd edn. Hanser, Munich Vienna (2012)

5. White, S.A., Miers, D.: BPMN Modeling and Reference Guide. Future Strategies, Lighthouse Point (2008)
6. Weske, M.: Business Process Management. Springer, Heidelberg (2007)
7. Dive Into Business Process Management, http://www.diveintobpm.org
8. Workflow Patterns, http://www.workflowpatterns.com
9. Ouyang, C., et al.: Workflow Management. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1, pp. 387–418. Springer, Heidelberg (2010)
10. Dugan, L., Palmer, N.: Making a BPMN 2.0 Model Executable. In: Fischer, L. (ed.) BPMN 2.0 Handbook, 2nd edn., pp. 71–91. Future Strategies, Lighthouse Point (2012)
11. Inubit Suite, http://www.inubit.com/en/inubit-suite.html
12. IYOPRO, http://www.iyopro.com/?lang=EN
13. van der Aalst, W., et al.: Business Process Simulation. In: vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1, pp. 313–338. Springer, Heidelberg (2010)
14. Januszczak, J.: Simulation for Business Process Management. In: Fischer, L. (ed.) BPMN 2.0 Handbook, 2nd edn., pp. 135–150. Future Strategies, Lighthouse Point (2012)
15. Waller, A., Clark, M., Enston, L.: L-SIM: Simulating BPMN Diagrams with a Purpose Built Engine. In: Perrone, L.F., et al. (eds.) Proceedings of the 2006 Winter Simulation Conference, pp. 591–597. IEEE, Piscataway (2006), http://www.informs-sim.org/wsc06papers/073.pdf
16. iGrafx Process (2011), http://www.igrafx.com/products/process
17. L-SIM Server for Business Process Simulation, http://www.lanner.com/en/l-sim.cfm
18. Silver, B.: BPMN Method & Style, 2nd edn. Cody-Cassidy Press, Aptos (2011)
19. van Dongen, B.F., Jansen-Vullers, M., Verbeek, H.M.W.E., van der Aalst, W.M.P.: Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. Comput. Ind. 58(6), 578–601 (2007)
20. Mendling, J.: Empirical Studies in Process Model Verification. In: Jensen, K., van der Aalst, W.M.P. (eds.) Transactions on Petri Nets and Other Models of Concurrency II. LNCS, vol. 5460, pp. 208–224. Springer, Heidelberg (2009)
21. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous Soundness Checking of Industrial Business Process Models. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 278–293. Springer, Heidelberg (2009)
22. Innovator for Business Analysts, http://www.mid.de/en/products/innovator-for-business-analysts
23. van Gorp, P., Dijkman, R.: BPMN 2.0 Execution Semantics Formalized as Graph Rewrite Rules: extended version. Beta Working Paper series 353. Eindhoven University of Technology (2011), http://cms.ieis.tue.nl/Beta/Files/WorkingPapers/wp_353.pdf
24. Sörensen, O.: Semantics of Joins in Cyclic BPMN Workflows. Diploma Thesis. Christian-Albrechts-University Kiel (2009), http://www.is.informatik.uni-kiel.de/~ove/research/papers/2009-OrJoin.pdf
25. zur Muehlen, M., Recker, J.: How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 465–479. Springer, Heidelberg (2008)
26. Signavio Process Editor, http://www.signavio.com