Information and Software Technology 55 (2013) 365–394

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A visual token-based formalization of BPMN 2.0 based on in-place transformations

Pieter Van Gorp *, Remco Dijkman

Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 20 December 2011
Received in revised form 29 August 2012
Accepted 31 August 2012
Available online 15 September 2012

Keywords:
BPMN
BPM
MDA
Formal semantics
Graph transformation

ABSTRACT

Context: The Business Process Model and Notation (BPMN) standard informally defines a precise execution semantics. It defines how process instances should be updated in a model during execution. Existing formalizations of the standard are incomplete and rely on mappings to other languages.
Objective: This paper provides a BPMN 2.0 semantics formalization that is more complete and intuitive than existing formalizations.
Method: The formalization consists of in-place graph transformation rules that are documented visually using BPMN syntax. In-place transformations update models directly and do not require mappings to other languages. We have used a mature tool and test-suite to develop a reference implementation of all rules.
Results: Our formalization is a promising complement to the standard, in particular because all rules have been extensively verified and because conceptual validation is facilitated (the informal semantics also describes in-place updates).
Conclusion: Since our formalization has already revealed problems with the standard and since the BPMN is still evolving, the maintainers of the standard can benefit from our results. Moreover, tool vendors can use our formalization and reference implementation for verifying conformance to the standard.

1. Introduction

The Business Process Model and Notation (BPMN) version 2.0 [41] is a standard notation for business process modeling. It presents a set of concepts and notational elements for business process modeling. It also presents an execution semantics that defines precisely how models in the BPMN notation should behave when executed in a tool. That semantics is defined informally using natural language.

There exist various initiatives to define a formal execution semantics in addition to the informal one [62,63,10,45,46,12,56]. These formal semantics are defined for a wide variety of reasons, including: enabling formal reasoning about the correctness of BPMN 2.0 process models, enabling simulation of those models and reasoning about the correctness of the standard. All of these formalizations rely however on a mapping (i.e., an out-of-place transformation [35]) of BPMN elements to elements in another formalism (such as Petri nets). This paper presents a formalization using visual, in-place [35], transformation rules. Defining the execution semantics using in-place transformation rules has three important benefits.

Fistly, the approach is intuitive since graph transformation rules can be defined by using the BPMN 2.0 notation itself.

Secondly, the approach is simple since there is good traceability between the informal execution semantics rules in the standard and their formal counterparts in a graph transformation form. This facilitates easy validation of the correctness of each of the formal rules. The traceability exists because the informal semantics in the standard is also defined in terms of a token-game. It implicitly relies on rules that specify when a certain notational element can be (de-)activated and what happens when it does. This can be mapped easily to a graph transformation rule, which always have a "match" part and a "rewrite" part. We have designed our rule set such that each notational element has two rules: one for activating the element and one for disabling the element. Regardless of the rule representation syntax (formal or informal, textual or visual), this consistent design already reduces complexity since the semantics of different elements can be considered in isolation.

Finally, using graph transformation rules allows the formalization to be complete. Theoretically, it is possible to develop a complete execution semantics of BPMN 2.0 in terms of graph transformation rules, because graph transformation is Turing complete [21]. This as opposed to, for example, classical Petri nets, in terms of which some constructs are notoriously hard to represent [10]. As a proof of this concept, our execution semantics covers more rules from the BPMN 2.0 standard than any other formal semantics so far.

* Corresponding author. Tel.: +31 40 2472062; fax: +31 40 2432612.
E-mail addresses: p.m.e.v.gorp@tue.nl (P. Van Gorp), r.m.dijkman@tue.nl (R. Dijkman).

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.08.014

There exists a wide variety of graph transformation tools that can execute graph transformation rules. Therefore, our execution semantics in terms of graph transformation rules can be verified on a test-suite of complex BPMN models. This has already enabled us to verify the expected behavior of our formalization. Moreover, we have discovered various points for improvement for the BPMN standard and this paper enables others to extend this work.

We therefore recommend the maintainers of the standard to (1) validate whether our formalization matches their intentions, (2) to use our supportive prototype as an instrument to improve the informal text in the standard and (3) to include an appendix based on our visual, rule-based formalization. This should lead to a better conformance to the standard and a better adoption of language constructs (especially non-trivial ones such as compensation events).

The remainder of this paper is structured as follows. Section 2 provides an introduction to graph transformation and BPMN 2.0. Section 3 defines the BPMN 2.0 execution semantics formally using graph transformation rules. Section 4 explains how we have constructed a reference implementation, how that implementation can be used and which alternatives we consider promising. Finally, Section 5 presents the evaluation of our contribution, Section 6 presents related work and Section 7 concludes.

2. Preliminaries

Before elaborating on the challenging topic of BPMN 2.0 semantics definition, Section 2.1 provides a gentle introduction to the BPMN. Section 2.2 then demonstrates our rule-based approach to semantics formalization on a language with a notoriously more simple semantics than the BPMN. That section serves as a "hello world" teaser to the next sections, which are more technically detailed. Section 2.3 provides a systematic introduction to typed attributed graphs and graph transformation. Finally, Section 2.4 introduces techniques for composing primitive rules into more powerful units.

2.1. BPMN 2.0

BPMN 2.0 can be used to create models of an organization's business processes. To this end, it defines a large number of notational elements, the meaning of those elements and an execution semantics that defines how certain combinations of elements should behave.

Fig. 1 shows a simple BPMN model of an order handling process. The model starts with a start event, represented by a circle, that is triggered when a message, represented by the envelope icon, arrives. The message contains an order. After the order arrives, the organization starts to process the order in a subprocess, represented by a rounded rectangle that contains other elements. The subprocess contains two activities, represented by rounded rectangles, and can be interrupted when a cancelation (represented by the cross symbol attached to the subprocess) about the order is received. After either the subprocess completes or an order cancelation is received, the alternative paths are joined by a so-called exclusive gateway, represented by the diamond with the "X". Finally, the process reaches an end event, representing completion.

The elements shown in Fig. 1 can be interpreted as a graph: the start event, end event, subprocess element, its contained activities, its attached message event and finally the exclusive gateway are the *nodes* of the graph while the flow arcs shown in Fig. 1 are its edges. Finally, for representing the hierarchical relation between the subprocess node and its children we can also assume the presence of a specificly typed edge between these elements.

The execution semantics of BPMN 2.0 is defined in terms of a large number of execution semantics rules. One of these rules, for example, states that the behavior of an exclusive gateway is such that: *"Each token arriving at any incoming Sequence Flows activates the gateway"*. We will introduce the graph-based formalization of this rule very gently in Section 2.3 but already demonstrate its expected behavior by means of Fig. 2. Note that the BPMN standard provides no standard icon for representing tokens, which makes it impossible to visualize process executions in standard syntax. We represent tokens as black dots, inspired by other flow-based languages such as classical Petri nets. Also note that Fig. 2 shows one process state per numbered item (1–14). Such a state consists of all process elements, all tokens, and all process instances to which these tokens belong. Again, inspired by Petri nets, we refer to these states as *Markings* and to the overall graph in Fig. 2 as a *statespace*.

Fig. 2 shows all possible executions of the order handling process (cfr., Fig. 1). The rule for executing exclusive gateways is triggered for transitioning from state 6 to 7 as well as from state 11 to 12. Clearly, for these transitions, the aforementioned behavior is satisfied. Note that the formal semantics from this paper enables the execution of BPMN models even when no guard expressions have been specified by the business analyst: all choices can be made non-deterministically or based on additional user input during process execution. We argue that this is quite valuable especially for business analysts, who are typically not trained in guard expression specification languages. For example when designing the process models from Fig. 1, the analyst may find it useful to double-check that regardless of arc inscription details there are two ways to complete the process and that in both final states (markings 9 and 14 from Fig. 2) there is exactly one token in the end event.

The figure also illustrates that upon order cancelation, tokens are removed from the subprocess elements as well as from the subprocess activity (cfr., the three transitions leading to state 10). Finally, note that although they are identical visually, state 7 is different from state 12 (and 8 from 13 and 9 from 14 respectively). More specifically, the state of the subprocess is "completed" for

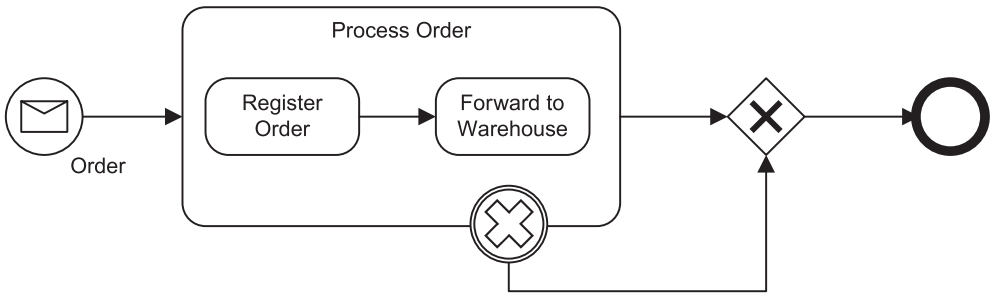

Fig. 1. Example BPMN 2.0 model: processing orders.