Abstract. We discuss ambiguities and inconsistencies in the Business Process Model and Notation (BPMN) 2.0 standard regarding the semantics of event-based gateways, and instantiating event-based gateways in particular. We suggest to use instantiating parallel event-based gateways to model asynchronous process behaviour and to clarify the BPMN standard accordingly. We further discourage from any other use of instantiating event-based gateways, and call for a clarification of the semantics of event-based gateways in general.

Keywords: business process modelling, BPMN, event-based gateway, instantiating event-based gateway, semantics.

1 Introduction

The authors have been working on a formalisation of the Business Process Model and Notation (BPMN) 2.0 standard [1] for some time. During this work, despite an intense study of the BPMN standard, we are still uncertain about the intended semantics of some BPMN elements. Often the BPMN standard specifies an element in a very general way in one place, and then constrains this description in various other places. After putting all the different descriptions of one element together, we have sometimes found apparent inconsistencies or even contradictions, while at the same time, the semantics of certain elements remains ambiguous. Studying further literature often confirmed that certain parts of the BPMN standard can be interpreted in different ways, while certain constructs seem to be ignored by the literature and by tools. Sometimes additional literature even added to our confusion.

In this paper, we want to discuss some of the problems we have encountered, namely those concerning event-based gateways, and instantiating event-based gateways in particular. More specifically, we discuss basic issues concerning the activation of events or receive tasks in an event-based gateway configuration, the point of time for creating process instances by instantiating event-based gateways, the instantiation of sub-processes by event-based gateways, and the meaning of ”waiting” for instantiating parallel event-based gateways.
After a short overview over related work in Section 2 we thoroughly discuss the issues mentioned above in Section 3 and sum our discussion and proposed solutions up in Section 4.

Please note that section numbers in square brackets, without an additional reference to a certain publication, always refer to the corresponding section in the BPMN standard.

2 Related Work

We did not find publications describing the semantics of all event-based gateway types specified in the BPMN standard [1]. However, amongst others, we found an approach by Dijkman et al. which uses a formal mapping from BPMN to Petri nets in order to be able to statically analyse business process models and check their semantic correctness [2]. The approach deals with a comprehensive subset of BPMN but does not consider instantiating event-based gateways. Cervantes also presents a mapping to Petri nets [3] where a mapping for event-based gateways is included, though not for instantiating event-based gateways. Several other formalisations of BPMN exist which do not regard event-based gateways.

Various different approaches exist for previous versions of the BPMN standard, e.g., a mapping to YAWL [4] or a mapping to BPEL [5]. Weidlich et al. present the other perspective of the latter alignment, i.e. a BPEL-to-BPMN mapping and its pitfalls [6]. For example, they show that the pick activity, with the attribute createInstance set to “no”, can be directly mapped to its counterpart in BPMN, the event-based gateway. However, they indicate compatibility issues concerning the process instantiation mechanisms of BPEL and BPMN, i.e., BPEL scenarios involving multiple start activities are only partially mappable to instantiating event-based-gateways.

Nicolae et al. apply a high-level modelling based on UML to provide a common understanding (in terms of an abstract syntax) of the involved concepts of Service Interaction Patterns, which are directly supported by BPMN 1.1 [7].

Russell et al. propose the deferred choice pattern [8] as one of their basic state-based patterns, where the moment of choice, i.e. the decision about which course of action should be taken, is deferred to a later time and based on external factors. (More specifically, there rather is a race between different event triggers or messages than an explicit choice.) This behaviour corresponds to the BPMN exclusive event-based gateway and does not consider issues such as process instantiation or waiting for multiple parallel branches. According to [8], the deferred choice pattern represents a complex pattern that, interestingly, seems to be successfully supported by token-based approaches. E.g., YAWL directly supports this pattern. In UML Activity Diagrams, fork nodes followed by interruptible activity regions can be applied to support deferred choice [9].

In [10] the racing incoming messages pattern is described using exclusive event-based gateways. Furthermore, the deferred choice pattern is extended by reaction rules attached to additional rule gateways in order to be able to constrain the decisions made by the environment. In doing so, the activation of activities can dynamically be determined using predefined conditions.