MIDA: Multiple Instances and Data Animator

Flavio Corradini, Chiara Muzi, Barbara Re, Lorenzo Rossi, and Francesco Tiezzi
School of Science and Technology, University of Camerino, Italy
name.surname@unicam.it

Abstract. The BPMN standard is largely adopted by industry and academia due to its intuitive graphical notation. Nevertheless, fully understanding the behaviour of BPMN collaboration models may be difficult when dealing at the same time with multiple instances, exchange of messages, and data manipulation. Figuring out the interplay between such concepts by statically looking at models is in general error-prone and time-consuming. To overcome this issue we provide a novel model animator tool, called MIDA. It turns out to be an effective supporting tool for *enhancing the understanding* of BPMN collaborations and *debugging* errors that can easily arise in modelling them.

Keywords: BPMN · Multi-Instance Collaborations · Data Handling · Animation

1 Introduction

The BPMN standard [6] is nowadays the most prominent modelling notation for business processes. In particular, BPMN collaboration diagrams provide an effective way to describe how multiple participants cooperate to reach a shared goal. However, the standard leaves unformalised the interplay between control flow, data handling and exchange of messages in scenarios requiring multiple instances of some interacting participants. These concepts are indeed strictly related to each other. The arrival of a *message* can *create a new process instance* if caught by a start message event, or can *drive the behaviour of a running instance* if caught during the process flow. In both cases, messages deliver data whose content can be used to fill *data objects* and to influence the process behaviour. In particular, the content of messages is used by the *correlation* mechanism to deliver them to the appropriate process instances. This is particularly useful in case of models including multi-instance pools. However, the lack of formalisation has consequently led to a lack of tools supporting designers in the modelling and debugging of collaborations involving these tricky, yet crucial, features of BPMN. In fact, designers are currently not assisted in figuring out the (possibly complex) behaviour of such models, and just looking at their static descriptions is in general an error-prone task.

To clarify this issue, we resort to a multi-instance collaboration model that will be used as a case study throughout this demo paper. The collaboration diagram depicted in Fig. 1 concerns with the management of a hotel booking involving three participants: the *Customer*, the *Booking System*, and the *Hotel*. The collaboration represents a scenario in which the *Customer* interacts with the *Booking System* in order to get a list of hotel quotes for a desired travel. To serve the customer request, the *Booking System* creates a process instance. This, in its turn, sends multiple requests (via a multi-instance sending task) to the *Hotel* pool, thus creating many instances of this latter participant. The

F. Casati et al. (Eds.): Proceedings of the Dissertation Award and Demonstration, Industrial Track at BPM 2018, CEUR-WS.org, 2018.

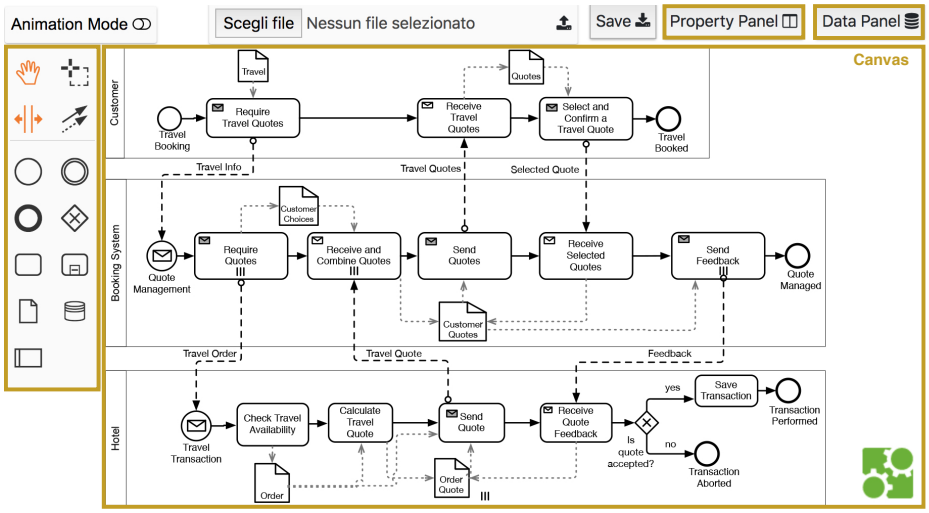

Fig. 1: Hotel Booking Collaboration.

Booking System collects the quotes for the *Customer* and sends them to him. Finally, the *Customer* communicates the selected hotel to the *Booking System*, which consequently gives a feedback to both selected and discarded *Hotel* instances. To model correctly this collaboration the diagram must contain enough information to correlate each message with the appropriate instance. For example, if messages are not properly delivered, an *Hotel* instance may save a transaction even if it has not been selected.

Model animation plays an important role to address the issues that may arise when modelling multi-instance collaborations, such as the intricate management of the correlation mechanism or the handling of data and messages. Animation can enhance the understanding of business processes behaviour [5,4], especially in presence of a faithful correspondence with a precise semantics [2] (which, in our case, is described in [3]). In the literature, few relevant contributions have been proposed: the animator by Allweyer and Schweitzer [1], and those developed by Signavio and Visual Paradigm. However, these tools do not support the interplay between multiple instances, messages and data, hence they do not allow designers to deal with the mentioned issues.

Therefore, we have developed MIDA (*Multiple Instances and Data Animator*), a novel animator tool. It supports designers in achieving a more precise understanding of the behaviour of a collaboration by means of the visualisation of the model execution, also in terms of the values evolution of data objects and messages. MIDA animation features result helpful both in *educational contexts*, for explaining the behaviour of BPMN elements, and in practical modelling activities, for *debugging* errors that can easily arise in multi-instance collaborations.

2 Main Features of MIDA

In this section, we present the MIDA modelling and animation features, specifically focussing on the interplay between data, messages and multiple instances.

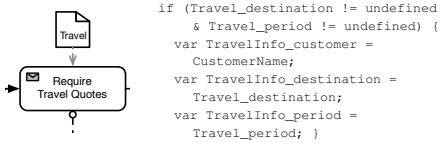

Fig. 2: Activity Guard and Assignments.

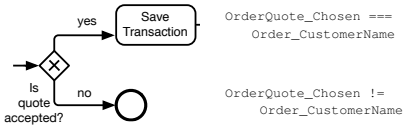

Fig. 3: XOR Conditions.

Modelling. MIDA is a web application written in JavaScript, accessible by users via a web browser without installing any plug-in or server backend. MIDA has been realised by extending the Camunda *bpmn.io* token simulation plug-in [7]. As shown in Fig. 1, the graphical interface of the tool consists of four main parts: *(i)* the canvas, where BPMN elements are composed to form a collaboration diagram; *(ii)* the palette, to insert elements in the diagram; *(iii)* the property panel, to specify attributes of the BPMN elements of the diagram; and *(iv)* the data panel, to visualise data object values. MIDA permits to locally save models in the standard format *.bpmn* and, hence, to load models previously designed.

The property panel plays a key role when modelling collaborations with MIDA, as it permits exploiting *.bpmn* XML attributes to model and save information about multi-instance characteristics, data objects and related fields, and messages. This information is written using the JavaScript syntax.

Both pools and activities can be set as multi-instance. In the former case, a double click on the pool element opens a pop-up window that allows to specify the pool as multi-instance and to constrain the number of instances that will be executed for that pool. In the latter case, multi-instance activities are defined by selecting the corresponding marker (||| or ≡) in the element context pad and by filling the *loopCardinality* attribute with the number of activity instances to execute.

Data objects are structured in terms of fields, which are rendered in MIDA as JavaScript variables that can be initialised or left undefined. According to the BPMN standard, the access to data is represented by associations between data objects and activities. These associations can define preconditions for the execution of an activity, expressed in MIDA as an activity *guard*. The effects on data objects of an activity execution is instead specified by means of a list of *assignments*. In case of send tasks, assignments can be used also to fill message fields while, in case of receive tasks, guards also specifies correlation conditions. For example, considering our running scenario, the guard and the assignments of the *Require Travel Quotes* task are expressed as reported in Fig. 2. Specifically, the condition of the `if` statement checks that the fields *destination* and *period* of the *Travel* data object are initialised. The content of such fields is then used by the task's assignments to fill the *Travel Info* message.

Values stored in data objects can be also used by conditions associated to the outgoing sequence flows of XOR split gateways, in order to drive choices. Concerning our case study, Fig. 3 reports the conditional expressions that check for a *Hotel* instance if its quote has been chosen or not by the customer, according to the content of the *Feedback* message it has received and stored in the *Order Quote* data object.

Animation. The key characteristic of MIDA is the animation of collaboration models, supporting in particular the visualisation of data evolution and multiple instances execution. At any time, the animation can be paused by the user to check the distribution of

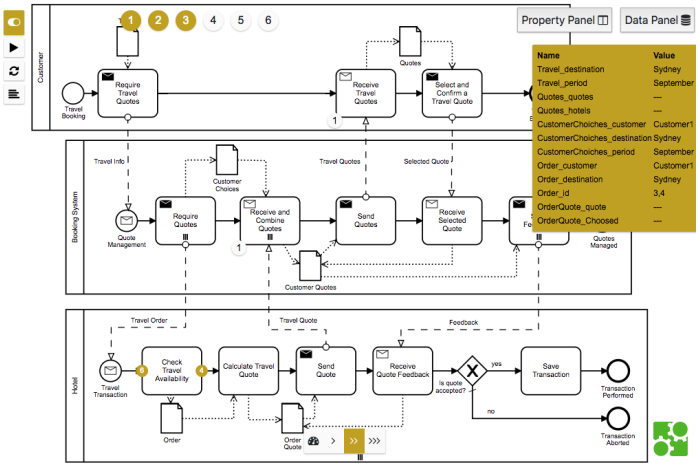

Fig. 4: MIDA Animation.

tokens and the current state of data. From a practical point of view, this allows designers to debug their collaboration models. They can indeed detect undesired executions, where e.g. a control flow is blocked, and deduce the cause beyond them by possibly checking the values of the involved data. These debugging facilities are particularly useful in case of multi-instance collaborations, where data values are used to regulate the correlation of messages with instances. Anyway, like in software code debugging, the identification and fixing of bugs are still in charge of the human user.

By selecting the *Animation Mode* in the MIDA interface, a *play* button will appear over each fireable start event. Once this button is clicked, one or more instances of the desired process are activated, depending on the multi-instance information specified in the model. This creates a new token labelled by a fresh instance identifier. Then, as shown in Fig. 4, the token starts to cross the model according to the operational rules induced by our formal semantics [3]. The animation terminates once all tokens cannot move forward. In case a token remains blocked due to a data handling issue, e.g. a wrong correlation or a guard condition violation, MIDA highlights it in red as in Fig. 5.

The *Data Panel* of the MIDA interface allows the user to monitor the evolution of data values. Fig. 6 shows how data values change after the execution of the task *Require Quotes*, which stores the values received via the *Travel Info* message in the fields *destination*, *period* and *customer* of the *Customer Choices* data object.

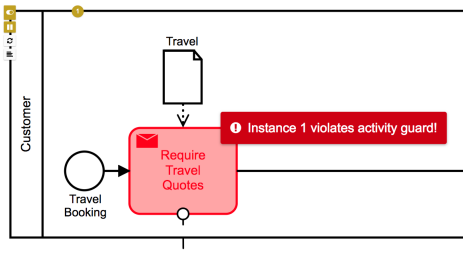

Fig. 5: Guard Violation.

3 Model Debugging with MIDA

In this section, we present how MIDA can effectively support designers in debugging their models. We resort to our case study in Fig. 1 to show how issues related to instance correlation can be detected.

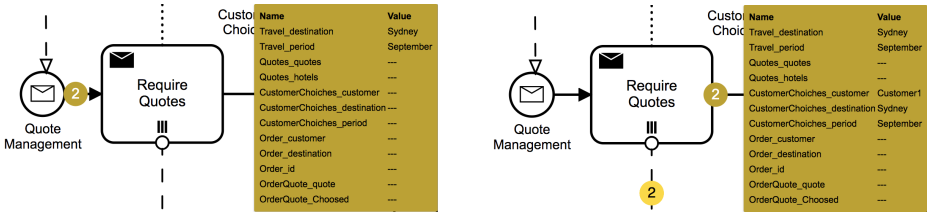

Fig. 6: Data Change after *Require Quotes* Task Execution.

Let us consider the effect of messages arrival into the *Hotel* pool. The arrival of a *Travel Order* message triggers the activation of a new process instance, while a *Feedback* message has to be routed to an already existing instance. Hence, in the latter case, the message needs to be properly correlated. In fact, after the choice performed by the *Customer*, only the selected *Hotel* instance has to receive a *positive Feedback* and then perform the *Save Transaction* task. To do that, each *Feedback* message contains the unique name of the *Hotel* that has to receive it. However, if the correlation check would be not properly specified in the receive task *Receive Quote Feedback* (e.g., only the travel period is used as correlation data or no correlation data is provided), the feedback messages would be not correctelly delivered. As consequence, the unwanted *Hotel* instances may book an overnight stay in vain, while the one selected by the *Customer* would not. This bug can be detected by MIDA as the *Hotel* chosen by the customer, stored in *Customer Quotes*, would not be the one stored in the *Order Quote* data object of the corresponding *Hotel* instance. To fix the bug, the designer can specify the conditional expression `OrderQuote_hotelName === Feedback_hotelName` of the receive task thus enabling the right message correlation.

4 Screencast and Website

The MIDA tool, as well as its source code, examples, user guide and screencast, are available at `http://pros.unicam.it/mida/`. In particular, the screencast shows a typical scenario where the user requires to model, animate, and debug a BPMN model. MIDA can be redistributed and/or modified under the terms of the MIT License.

References

1. Allweyer, T., Schweitzer, S.: A tool for animating BPMN token flow. In: BPMN Workshop. LNBIP, vol. 125, pp. 98–106. Springer (2012)
2. Becker, J., Kugeler, M., Rosemann, M.: Process management: a guide for the design of business processes. Springer Science & Business Media (2013)
3. Corradini, F., Muzi, C., Re, B., Rossi, L., Tiezzi, F.: Animating Multiple Instances in BPMN Collaborations: from Formal Semantics to Tool Support (2018), BPM'18 - To Appear.
4. Desel, J.: Teaching system modeling, simulation and validation. In: WSC. pp. 1669–1675 (2000)
5. Hermann et al., A.: Collaborative business process management - a literature-based analysis of methods for supporting model understandability. In: WI (2017)
6. OMG: Business Process Model and Notation (BPMN V 2.0) (2011)
7. Philipp Fromme and Sebastian Warnke and Patrick Dehn: bpmn-js Token Simulation (2017), `https://github.com/bpmn-io/bpmn-js-token-simulation`