The Shared Process Model

Cédric Favre, Jochen Küster, and Hagen Völzer

IBM Research - Zurich, Switzerland,
{ced,jku,hvo}@zurich.ibm.com

Abstract. Maintaining different but consistent views on the same process is often necessary in BPM projects. For example, a business analyst typically works on a business process model at a different level of abstraction than an IT architect. These views evolve independently and synchronizing them is not trivial. In this demonstration, we showcase our Shared Process Model prototype that allows different stakeholders to work on different views of a business process model while keeping these views synchronized. In particular, we will look at scenarios where a business view and an IT view are modified and a subset of the modifications need to be propagated from one view to the other. This demonstration targets the general BPM audience interested in ensuring consistency between various level of realisation of a business process model and solving the related round tripping problems. This demonstration will also appeal to people interested in process comparison and process merging — the two core techniques used by our prototype to propagate changes from one view to the other.

1 Relevance to BPM field

A business process model is used by different stakeholders for different purposes. For example, a business analyst uses a business process model to document, analyze and communicate a process while an IT architect uses a process model to implement the process on a particular process engine. Both stakeholders use a model that represents the same process but from a different perspective which has different requirements. For example, the IT architect is interested in modeling the service invoked when a task is executed and the exception flow triggered when the service invocation fails. For the business analyst, these implementation details clutter the process and therefore should not appear in his view. Note that the differences are not only IT refinements, i.e., the business view is not just a subset of the IT view. We study the differences between the two perspectives in more detail elsewhere [4].

In this demo, we will showcase our *Shared Process Model* prototype. The Shared Process Model supports parallel maintenance of different views of a the same BPMN 2.0 model, a capability lacking in major BPM suites [2]. In Sect. 2, we discuss the features, the supported scenarios, and an overview of the implemented approach of our prototype. In Sect. 3, we describe a scenario that we will use as screen-cast to highlight the features of the Shared Process Model. In Sect. 4, we conclude with the limitations of the prototype and future work.

2 The Shared Process Model

The Shared Process Model is a research prototype built on top of the *BPMN2 Modeler* — an Eclipse-based graphical BPMN 2.0 model editor [1]. The Shared Process Model provides to the users two different views on a process, a *business view* and an *IT view* as illustrated by Fig. 1.

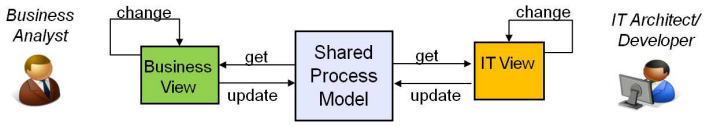

Fig. 1: The Shared Process Model

Supported operations: The Shared Process Model supports the independent development of a business-view and an IT view of a process through tree key operations:

get returns a copy of the current version of the IT or business view,
change allows the user to modify his copy of the IT or business view, and
update allows the user to synchronize his copy of the IT or business view with the Shared Process Model. Each change can be either designated as a *common* or a *private* change. A common change is automatically propagated by the Shared Process Model to the other view whereas a private change is not.

Central features: The Shared Process Model allows a user to modify either the business or the IT view and to propagate a subset of these modifications to the other view. For example, the IT architect might decide to update changes made to the main control-flow of the process as common but to keep private the addition of the exception flow. The IT architect may need to propagate changes to the business view because the initial business model is incomplete, contains modeling errors, contradicts some IT requirements, or does not faithfully represent the actual business process. Propagating changes is also required when the business or the IT requirements change. The reasons and frequency of these updates are presented in more detail in a technical report [4].

It also ensures that the two views remain *consistent*, i.e., the IT view is a 'faithful' representation of the business view and vice et versa. The enxact notions of consistency considered ad how they are ensured or checked is out of scope of this paper but are presented in the technical report [4].

Finally, the Shared Process Model provides a set of model *refactoring operations* to support the user in modifying a view while retaining its consistency with the other view. For example, it provides refactoring operations to refine an activity into a subprocess or into a set of activities together with the control-flow between them, to specify that an activity is implemented as a script task or a service task, and to simplify a portion of the process into a single activity.

Shared Process Model implementation and change propagation: Fig. 2 illustrates the internal of our implementation a Shared Process Model: two BPMN 2.0 process models representing the two views and *correspondences* (the highlighted vertical arrows) relating the nodes of the two models. Note that correspondences can point to multiple corresponding nodes and that some nodes of the IT view do not have a corresponding node in the business view.

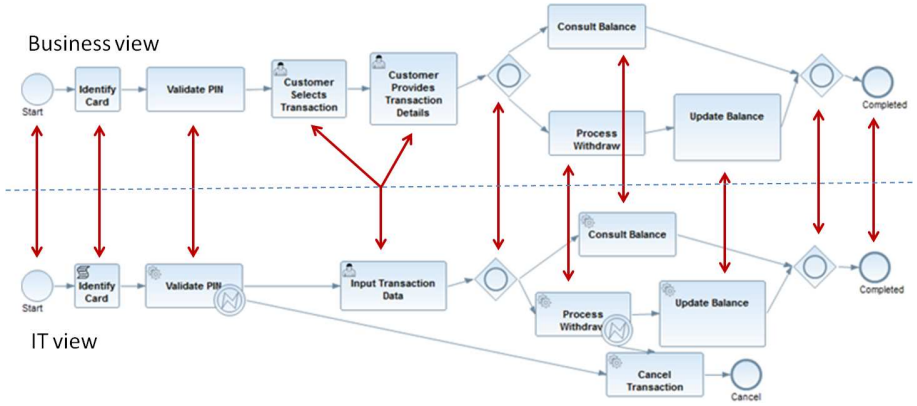

Fig. 2: A example of Shared Process Model internals

When a user modifies one view, let us say the IT view, and updates the Shared Process Model, we compute the changes between the updated IT view and the Shared Process Model version of the IT view (which represent the view before the modifications) using a comparison framework [5]. The common changes, formulated for the IT view, are translated into changes applicable to the business view. The translation is based on the correspondences and, when navigating correspondences with multiple targets in the business view, it uses structural analysis on the business view. Finally, the common changes are merged using a merging framework that we developed.

3 The Shared Process Model in Action

We now present a short scenario where the Shared Process Model is used to synchronize a business and an IT view on a process. A screen-cast of featuring this scenario is available on the project webpage [3]. This scenario features two actors Alan, a business analyst and Paul, an IT architect.

Initialization of the Shared Model First, Alan captures the business process model illustrated by Fig. 3. Alan initializes the Shared Process Model, which now contains this process in both views and the appropriate correspondences. From now on, Alan will work on the business view and Paul on the IT view. Alan asks Paul to create an implementation of the process model.

Fig. 3: Initial business process model

Private and Common Changes: Using refactoring operations, Paul refines the specification of the activities by specifying their realization, some activities are implemented as script tasks other by service tasks. He also adds the exception flow. These changes are only relevant to the IT view. Therefore, Paul commits the changes as private. Paul then realizes that he can optimize the control-flow of the process and that Alan forgot one activity. These changes are relevant to business view and Paul updates them as common. The IT view now displays the process in Fig. 4 while the business view displays the process illustrated on top of Fig. 2.

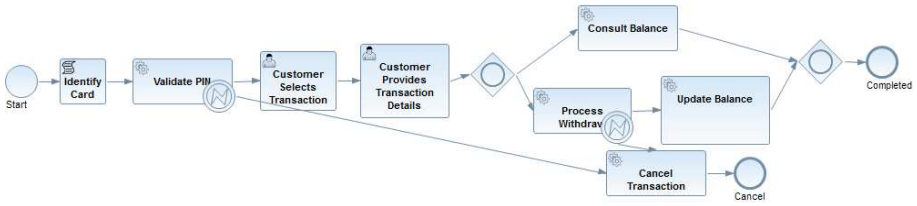

Fig. 4: IT view after private and common changes

Private refactoring: Paul wants to convert the two human activities into one. He uses the *simplify selection* refactoring wizard, which turns a selection of elements into a single activity and reroutes the incident edges of the selection accordingly. The refactoring also creates the correspondence between the two business activities and the IT activity. This change is a private change. The Shared Process Model now contains the two views illustrated by Fig. 2.

Common changes using correspondences Finally, Paul inserts a new activity between 'Validate Pin' and 'Input Transaction Data' as a common change which results in the IT view illustrated by Fig. 5. Looking at Fig. 2, one realizes that the translation of this change, as described in Sect. 2, requires to navigate the correspondences and to perform a structural analysis.

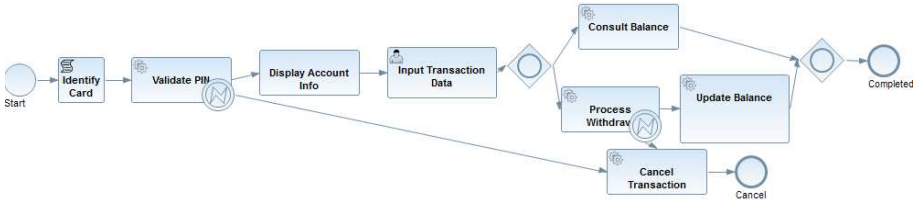

Fig. 5: IT view after selection simplification and insertion of task

Updating this change results in the business view illustrated by Fig. 6. The two views have evolved and are now significantly different. However, the Shared

Process Model is still able to propagate automatically changes between the two views ensuring that they stay consistent maintaining the correspondences between its different elements.

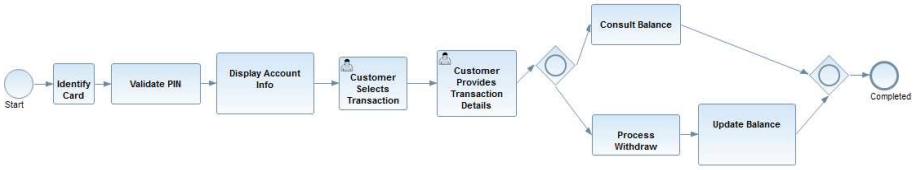

Fig. 6: Final business view

4 Limitations and future work

In this prototype, we focused on the propagation of control-flow related modifications. The propagation of some attribute modifications such as, for example, changing the type of an event is not implemented but could easily be added. We also consider the implementation of a user interface allowing a user to approve or reject the changes made by another user. For example, Alan could approve only a subset of the common changes proposed by Paul.

We only presented the modifications of two views: business and IT. The Shared Process Model can be generalized to any two BPMN 2.0 views. The architecture of the prototype as well as comparison and merging components would scale up to a larger number of views. However, the correspondences would require a more complex representation and management.

We currently support scenarios where IT view and business view modifications are interleaved. Ultimately, we aim to integrate the Shared Process Model in a modern BPM suite where models sit in a shared repository, the Shared Process Model would then be a shared object in the repository and new scenarios would involve concurrent editing of the IT and business view. Support to prevent, detect, and resolve conflicts arising from concurrent editing is necessary for these scenarios.

References

1. The BPMN 2.0 Modeler available at `http://eclipse.org/projects/project.php?id=soa.bpmn2-modeler`
2. M. Castelo Branco, K. Czarnecki, J. Kuester, and H. Völzer. An Empirical Study on Consistency Management of Business and IT Process Models. available at `http://gsd.uwarterloo.ca/reportstudybpm`
3. C. Favre, J. Kuester, and H. Völzer. The Artifact Consistency Management project: `http://researcher.watson.ibm.com/researcher/view_project.php?id=3210`
4. J. Kuester, H. Völzer, C. Favre, M. Castelo Branco, and K. Czarnecki. Supporting Different Process Views through a Shared Process Model. Technical report, IBM Research, RZ3823.
5. J. Küster, C. Gerth, A. Förster, and G. Engels. Detecting and resolving process model differences in the absence of a change log. *Business Process Management*, pages 244–260, 2008.