Supporting Business and IT through Updatable Process Views: The *proView* Demonstrator

Jens Kolb and Manfred Reichert

Ulm University, Germany
{jens.kolb,manfred.reichert}@uni-ulm.de
http://www.uni-ulm.de/dbis

Abstract. The increasing adoption of process-aware information systems (PAISs) has resulted in large process model collections. To support business and IT users having different perspectives on processes, a PAIS should provide personalized views on process models. Especially, changing process models is frequent use case due to evolving processes or unplanned situations. This demonstration presents the *proView* framework for changing large process models through updating process views, while ensuring up-to-dateness and consistency of all related process views. More precisely, update operations can be applied to a process view and are correctly propagated to the underlying process model. Further, all views related to this process model are then correctly migrated to its new version. Overall, the *proView* framework enables users to evolve process models over time based on appropriate model abstractions.

1 Introduction

Process-aware information systems (PAISs) separate process logic from application code relying on explicit *process models* [1]. This separation of concerns increases maintainability and reduces costs of change. The increasing adoption of PAISs has resulted in large process model collections. In turn, each process model may refer to different domains and user roles as well as dozens or even hundreds of activities. Usually, the different user roles (e.g., business or IT) need customized views on process models, enabling personalized process abstraction and visualization [2]. For example, managers rather prefer an abstract overview, whereas the IT department needs a detailed view of process parts implemented by a PAIS [3]. Hence, providing personalized process views is a much needed feature. Existing approaches for creating process abstractions, however, do not consider *change* and *evolution*, which are fundamental for PAISs [4]. In addition to view-based process abstractions, users should be allowed to change large process models through updating respective process views. However, this must not be accomplished in an uncontrolled manner to avoid inconsistencies or errors.

The *proView*[1] framework provides powerful view-creation operations [5]. The operations allow abstracting process models through *reduction* and *aggregation* of process elements. In addition, update operations allow adapting process views

[1] http://www.dbis.info/proView

A. Ghose et al. (Eds.): ICSOC 2012, LNCS 7759, pp. 460–464, 2013.

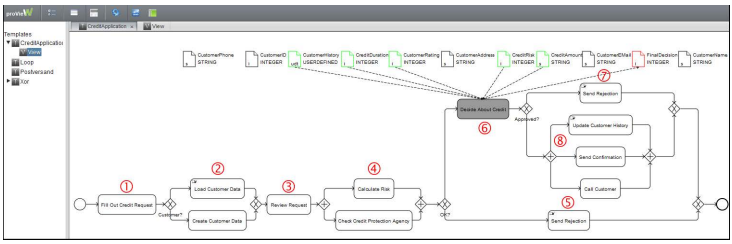

Fig. 1. Credit Application Process

and propagating the changes to the underlying process model as well as to other related process views [6]. We will demonstrate these aspects of the *proView* framework in an integrated way.

Section 2 introduces a scenario. Section 3 sketches the *proView* framework and the view operations it supports. Section 4 describes how the scenario can be supported by using the *proView* framework. Section 5 concludes the paper.

2 Application Scenario

Figure 1 shows a credit request process modeled in terms of BPMN. The process involves human activities referring to three user roles (i.e., *customer*, *clerk*, and *manager*) as well as automated activities (i.e., services) executed by the PAIS. Assume that the process is started by the customer filling out a credit request form (Step ①). Afterwards, the PAIS checks whether for this customer an entry in the CRM exists (Step ②). In this case, customer data is retrieved. Then, the clerk reviews the credit request (Step ③), calculates the risk, and checks creditworthiness (Step ④). After completing these tasks, he decides whether to reject the request (Step ⑤) or to forward it to his manager who decides about the request (Step ⑥). If the manager rejects, an email is sent (Step ⑦). Otherwise, a confirmation email is sent, the CRM is updated, the clerk calls the customer (Step ⑧), and the process completes. Assume that a process change is required: Before filling out the credit form, the customer shall select the desired credit type. For this purpose, the clerk adds an activity to the process model. Obviously, this change is relevant for all participants.

The *proView* framework addresses such a user-centered visualization and adaptation of large process models. Hence, it enables personalized views of the credit request process for each user role, i.e., customer, clerk, and manager. The following requirements must be met in order to properly support such a scenario:

- R1: For each user role, it should be possible to provide specific process views on a process model as well as flexibly defining those views.
- R2: Based on personalized process views and visualizations, elementary model adaptations shall become possible, e.g., to insert or delete activities in a user-centered process model (i.e., process view).
- R3: If an authorized user changes his process view other process views may have to be updated to ensure up-to-dateness of all process participants.

R4: Since domain experts hardly have technical process knowledge, high-level operations for creating and adapting user-centered process views are required.

3 proView Framework

Figure 2 gives an overview of the implemented *proView* framework, which consists of two major components: *proViewServer* and *proViewClient*. The *proViewClient* is instantiated for each user and handles the interactions with the user as well as the visualization of his process models and views. In turn, the proViewClient is based on the *vaadin* web-framework and interacts with the proViewServer using a RESTful protocol. The *proViewServer* implements the logic of the *proView* framework and provides engines for *visualization*, *change*, and *execution & monitoring* [7]. It captures a *business process* through a *Central Process Model (CPM)*; additionally, so-called *creation sets (CS)* are defined, with, each CS specifiying the schema and appearance of a process view [6].

The *visualization engine* generates a specific process view based on a given CPM and creation set CS, i.e., the CPM is transformed to the view by applying the *view-creation operations* specified in CS (Step ⑤). Afterwards, the obtained view is *simplified* by applying well-defined *refactoring operations* (Step ⑥). Finally, Step ⑦ customizes the visual appearance of the view; e.g., by creating a tree-, form-, or flow-based visualization).

When a user updates a view, the *change engine* is triggered (Step ①). First, the view-based model change is propagated to the CPM using well-defined propagation algorithms (Step ②). Next, the CPM is simplified (Step ③), i.e., behaviour-preserving refactorings are applied to foster model comprehensibility (e.g., by removing gateways not needed anymore). Afterwards, the creation sets of all views associated with the CPM are migrated to the new CPM version (Step ④). This becomes necessary since a creation set may be contradicting with the changed CPM. Finally, all views are recreated (Steps ⑤-⑦).

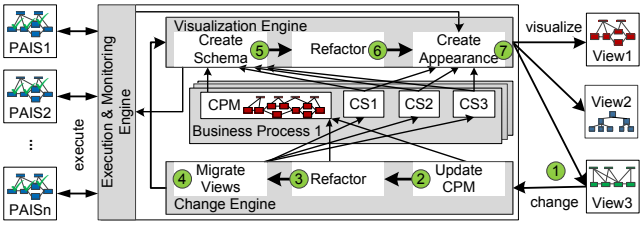

Fig. 2. The proView Framework

4 proView Demonstration

We revisit our scenario from Section 2 and show how the described requirements can be addressed by the *proView* framework.

Requirement R1: The proViewServer allows creating an arbitrary number of process views by applying aggregation and reduction operations specified in a creation set; e.g., a *reduction* removes an activity from the respective view, while an *aggregation* combines a set of connected activities to one node.

Requirement R2: The proViewServer provides *view-update* operations which allow inserting and deleting activities as well as AND/XOR branchings [6]. These operations can be applied by an end-user to his process view. Furthermore, *parametrization* allows for automatically resolving ambiguities when propagating view changes; i.e., change propagation behaviour can be customized.

Requirement R3: Updates triggered by users are applied to the CPM as well as to associated process views. Their view creation sets are then migrated to the new version of the CPM. Hence, all affected views will be re-created.

Requirement R4: The proViewServer supports high-level operations to create process views. For example, one may create a view can showing all technical activities or activities of a specific user role.

All these aspects are illustrated in a screencast: www.dbis.info/proView.

5 Conclusion

In our demonstration, we present the *proView* framework and its operations; *proView* supports the creation of personalized process views as well as the view-based change of business processes, i.e., process abstractions not only serve visualization purpose, but also lift process changes up to a higher semantical level. A set of change operations enables users to update their view. The respective change is then propagated to the CPM representing the overall business process and other process views are migrated to the CPM.

The *proView* framework is implemented as a client-server application to concurrently edit a process model based on views. The implementation of the proView framework has proven the applicability of our approach. Further, the *proView* demonstrator shall be extended to also execute process views in a PAIS. Overall, we believe that the *proView* framework offers promising perspectives to users for evolving their business processes.

References

1. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems - Challenges, Methods, Technologies. Springer (2012)
2. Rinderle, S., Bobrik, R., Reichert, M., Bauer, T.: Businesss Process Visualization - Use Cases, Challenges, Solutions. In: Proc. ICEIS 2006, pp. 204–211 (2006)
3. Bobrik, R., Reichert, M., Bauer, T.: Requirements for the Visualization of System-Spanning Business Processes. In: Proc. DEXA 2005 Workshops, pp. 948–954 (2005)
4. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. Data & Knowledge Engineering 66(3), 438–466 (2008)

5. Reichert, M., Kolb, J., Bobrik, R., Bauer, T.: Enabling Personalized Visualization of Large Business Processes through Parameterizable Views. In: Proc. 26th Symposium on Applied Computing (SAC 2012), Riva del Garda (Trento), Italy (2012)
6. Kolb, J., Kammerer, K., Reichert, M.: Updatable Process Views for User-Centered Adaption of Large Process Models. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) ICSOC 2012. LNCS, vol. 7636, pp. 484–498. Springer, Heidelberg (2012)
7. Kolb, J., Hübner, P., Reichert, M.: Automatically Generating and Updating User Interface Components in Process-Aware Information Systems. In: Proc. 10th Int'l Conf. on Cooperative Information Systems (CoopIS 2012), pp. 444–454 (2012)