# Updatable Process Views
# for Adapting Large Process Models:
# The *proView* Demonstrator

Jens Kolb, Klaus Kammerer and Manfred Reichert

Institute of Databases and Information Systems
Ulm University, Germany
{jens.kolb,klaus.kammerer,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 users having different perspectives on these processes and related data, a PAIS should provide personalized views on process models. Especially, changing process models is a frequent use case in PAISs due to evolving business processes or unplanned situations. While process views have been suggested as abstractions for visualizing large process models, no work exists on how to change these models based on respective views. This software demonstration presents the *proView* framework for changing large process models through updates of corresponding process views, while ensuring up-to-dateness and consistency of all other process views related to the changed process model. Respective update operations can be applied to a process view and are correctly propagated to the underlying process model. Furthermore, all views related to this process model are then correctly migrated to its new version as well. Overall, the *proView* framework enables domain experts to evolve large process models over time based on appropriate model abstractions.

**Keywords:** process model abstraction, process view, process change, view update, process visualization, user-centered process management

## 1   Introduction

Process-aware information systems (PAISs) provide support for business processes at the operational level [1]. A PAIS strictly separates process logic from application code, relying on explicit *process models*. This enables a separation of concerns, which is a well-established principle in computer science to increase maintainability and to reduce costs of change [2]. The increasing adoption of PAISs has resulted in large process model collections. In turn, each process model may involve different domains, organizational units, and user roles as well as dozens or even hundreds of activities [3]. Usually, the different user roles need customized views on their process models, enabling personalized process

abstraction and visualization [4,5]. For example, managers rather prefer an abstract overview, whereas process participants need a detailed view of the process parts they are involved in [6]. Hence, providing personalized process views is a much needed PAIS feature. A variety of approaches for creating process model abstractions based on process views have been proposed [7,8,9,10]. However, these proposals focus on creating and visualizing views, but do not consider another fundamental aspect of PAISs: change and evolution [11]. More precisely, they do not allow changing a large process model through editing or updating any of its view-based abstractions. As a consequence, process changes still must be directly applied to the core process model, which constitutes a complex as well as error-prone task for domain experts, particularly when confronted with large process models [12]. To overcome this limitation, 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 addresses these challenges by providing powerful view-creation operations [13]. The operations allow abstracting process models through the *reduction* and *aggregation* of process elements as well as through changes of the process model notation [14]. In addition, view-update operations allow adapting process views and propagating the respective changes to the underlying process model as well as to other related process views [15]. Our tool presentation will demonstrate these aspects of the *proView* framework in an integrated and comprehensible way.

Section 2 introduces the application scenario we use for our demonstration. Section 3 presents the *proView* framework and the view operations it supports. Section 4 then describes how the application 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 automatic activities executed by the PAIS without user interaction. Assume that the process is started by the customer filling out a credit request form (Step ①). Afterwards, the PAIS checks whether an entry for the customer needs to be created in the CRM system or the customer has been already registered (Step ②). In the latter case, customer information is retrieved from the CRM. Then, the clerk reviews the credit request (Step ③), calculates the risk, and checks the creditworthiness of the customer with the credit protection agency (Step ④). After completing these tasks, he decides whether to reject the request (Step ⑤) or forward it to his manager who finally decides about granting the credit request or not (Step ⑥). If the manager rejects the request, a respective email is sent to the customer (Step ⑦). Otherwise, a confirmation email is sent and the CRM database is updated. Finally, the clerk calls
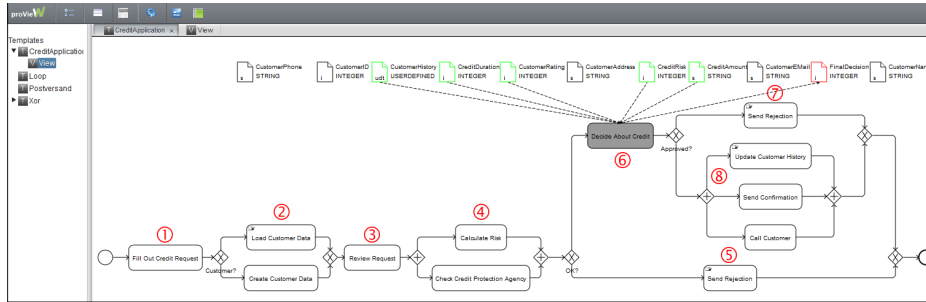
---

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

**Fig. 1.** Credit Application Process

the customer in the context of after sales (Step ⑧), before the process completes. Assume that an evolution of this process model becomes necessary: Before filling out the credit form, the customer shall select the desired credit type. For this purpose, an activity is added by the clerk to the process model. Obviously, this change is relevant for all participants.

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

R1: It should be possible to provide specific process views on a process model for each user role and to flexibly define those views.

R2: The visual appearance of the process model and process view respectively needs to be flexibly adaptable for each user (role) to meet needs best.

R3: Based on personalized process views and visualizations, elementary model adaptations should be possible, e.g., to insert or delete activities in a user-centered process model (i.e., process view).

R4: In case of changes introduced by a user, all other process views need to be updated to ensure up-to-dateness of all process participants.

R5: 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 *proView-Client* is instantiated for each user and takes care of interactions with the user as well as the visualization of his process models and process views respectively. The proViewClient is based on the *vaadin* web-framework and interacts with the proViewServer using a RESTful communication protocol. The *proViewServer*

implements the logic of the *proView* framework and provides engines for *visualization*, *change*, and *execution & monitoring*. It captures a *business process* through a *Central Process Model (CPM)*. In addition, for a particular CPM, so-called *creation sets (CS)* are defined. Thereby, each CS specifies the schema and appearance of a particular process view [15].

The *visualization engine* generates a process view based on a given CPM and the information captured in a creation set CS, i.e., the CPM schema is transformed to the view schema by applying the corresponding *view-creation operations* specified in CS (Step ⑤). Afterwards, the obtained view schema is *simplified* by applying well-defined *refactoring operations* (Step ⑥). Finally, Step ⑦ customizes the visual appearance of the view (e.g., creating an tree-, form-, or activity-based visualization [8,14]) and delivers it to the proViewClient.
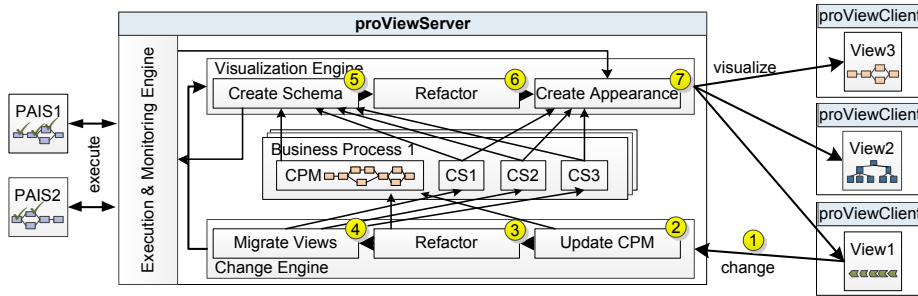


**Fig. 2.** The proView Framework

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

## 4 proView Demonstration

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

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

*Requirement R2*: The proViewClient enables users to change the visual appearance of process views, e.g., by switching between the notations provided by

BPMN, ADEPT [16], and proViewForms. The latter allow visualizing process models and views in terms of forms, which support users, not familiar with activity-centered process notations, in understanding complex process logic. Further visual appearances for process views are under construction (e.g., text-based representation).

*Requirement R3*: The proViewServer provides *view-update* operations which allow inserting and deleting activities as well as AND/XOR branchings [15]. These operations can be applied by an end-user to his process view using the proViewClient and are then be propagated to the proViewServer. Furthermore, *parametrization* of these operations allows for automatically resolving ambiguities when propagating view changes; i.e., change propagation behaviour can be customized. However, at this stage concurrent changes are not enabled in the proViewServer, i.e., only one change at a time is allowed.

*Requirement R4*: 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 R5*: The proViewServer supports high-level operations to create process views. For example, a new view can be created based on the role of a user displaying only those activities he is involved in.

All these aspects are illustrated in our screencast and can be watched at the projects' website: 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 update operations enables users to update their view and propagate the respective change to the process model representing the overall business process. Finally, we provide migration rules to update all other process views associated with a changed CPM. Similar to this propagation, it can be decided per view, how much information about the change shall be displayed to the user.

The *proView* framework is implemented as a client-server application to simultaneously edit a process model based on views. The implementation of the proView framework has proven the applicability of our approach. Furthermore, user experiments based on the proView demonstrator are planned to systematically analyze whether view-based process changes improve the handling and evolution of large process models. Moreover, the proView demonstrator shall be extended to also execute process views in a PAIS [17]. Overall, we believe that the *proView* framework offers promising perspectives for process participants for evolving their business processes.

# References

1. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems - Challenges, Methods, Technologies. Springer (2012)
2. Weber, B., Sadiq, S., Reichert, M.: Beyond Rigidity - Dynamic Process Lifecycle Support: A Survey on Dynamic Changes in Process-Aware Information Systems. Computer Science - Research and Development **23**(2) (2009) 47–65
3. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring Large Process Model Repositories. Computers in Industry **62**(5) (2011) 467–486
4. Rinderle, S., Bobrik, R., Reichert, M., Bauer, T.: Businesss Process Visualization - Use Cases, Challenges, Solutions. In: Proc. 8th Int'l Conf. on Enterprise Information Systems (ICEIS'06). Volume 2006., Paphos, Cyprus (2006) 204–211
5. Streit, A., Pham, B., Brown, R.: Visualization Support for Managing Large Business Process Specifications. In: Proc. BPM'05. (2005) 205–219
6. Bobrik, R., Reichert, M., Bauer, T.: Requirements for the Visualization of System-Spanning Business Processes. Proc. DEXA'05 Workshops (2005) 948–954
7. Tran, H.: View-Based and Model-Driven Approach for Process-Driven, Service-Oriented Architectures. TU Wien, Dissertation (2009)
8. Bobrik, R., Bauer, T., Reichert, M.: Proviado - Personalized and Configurable Visualizations of Business Processes. In: Proc. EC-WEB'06. (2006) 61–71
9. Chiu, D.K., Cheung, S., Till, S., Karlapalem, K., Li, Q., Kafeza, E.: Workflow View Driven Cross-Organizational Interoperability in a Web Service Environment. Information Technology and Management **5**(3/4) (July 2004) 221–250
10. Bobrik, R., Reichert, M., Bauer, T.: View-Based Process Visualization. In: Proc. 5th Int'l Conf. on Business Process Management, Brisbane, Australia (2007) 88–95
11. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. Data & Knowledge Engineering **66**(3) (2008) 438–466
12. Reijers, H., Mendling, J.: A Study into the Factors that Influence the Understandability of Business Process Models. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on (99) (2011) 1–14
13. 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'12), Riva del Garda (Trento), Italy (2012)
14. Kolb, J., Reichert, M.: Using Concurrent Task Trees for Stakeholder-centered Modeling and Visualization of Business Processes. In: Proc. S-BPM ONE 2012, CCIS 284. (2012) 237–251
15. Kolb, J., Kammerer, K., Reichert, M.: Updatable Process Views for User-centered Adaption of Large Process Models. In: Proc. Intl. Conf. on Service Oriented Computing (ICSOC'12), Shanghai, China (2012) to appear
16. Dadam, P., Reichert, M.: The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support. Computer Science - Research and Development **23**(2) (April 2009) 81–97
17. 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'12). (2012) to appear