A Runtime Environment for Object-Aware Processes

Kevin Andrews, Sebastian Steinau, and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Germany
{kevin.andrews,sebastian.steinau,manfred.reichert}@uni-ulm.de

Abstract. In contrast to contemporary activity-centric process-aware information systems (PAIS), for which a multitude of concepts and implementations exist, there is only a very limited number of PAIS implementations using data-centric, artifact-centric or object-aware approaches. This demo paper presents the implementation of a client-server runtime environment for the object-aware approach to process management. Our implementation is based on the PHILharmonicFlows conceptual framework, where individual processes define the behavior of an object and its interactions with other objects. The current implementation of the runtime environment allows for the instantiation and execution of *micro processes*, which define object behavior. Interaction with a data-driven micro process instance is enabled through automatically generated user-forms as part of a graphical user interface. Additionally, the user interface can display the progression of a micro process instance using an interactive graph.

1 Introduction

Hard-coding the multitude of forms necessary for modern business applications, as well as their respective internal logic, makes developing or changing business applications resource- and time-consuming. Furthermore, increasing competitive pressure and reduced time-to-market require companies to develop business applications at higher speeds without letting quality deteriorate.

PHILharmonicFlows is a framework for object-aware process management that aims at reducing the development and maintenance time of forms in regard to changes in the data model or individual business processes [1,3,4]. In object-aware processes, data is organized as objects which exhibit a defined behavior. The behavior specifies what data is required at which point in time, as well as the data semantics and interdependencies, e.g., taking cardinalities of semantic relationships into account. A *micro process* is used to define an object's behavior [2]. As in many business applications or case handling systems, the primary means for PHILharmonicFlows processes to acquire data are forms. The internal logic of each form, as well as the decisions, which forms and form fields need to

be displayed at which point in time, is governed by an object's micro process. Since the aim is to avoid the hard-coding of forms, the micro process concept was designed in a way that permits the automatic generation of forms from the micro process model.

The tool presented in this demonstration allows for the execution of micro processes and demonstrates the corresponding automatic form generation.

2 PHILharmonicFlows - The Micro Process Perspective

As mentioned in Section 1, PHILharmonicFlows allows for the generation of generic user forms. To this end, the framework introduces the concept of an *object* as a collection of atomic data types, called *attributes*. The attributes belonging to an object define all possibly displayable form fields concerning the real world entity the object represents, such as a job application. Clearly, for one real world entity, multiple forms become necessary. The forms must be shown to the various process participants at different points in time. A job application, for instance, must be created by an applicant, using a form. At a later point in the execution of the process, an application reviewer must be shown a different form, which allows him to see and rate the information that the applicant entered.

In order to support multiple forms per object, the object's attributes can be grouped into *states*. Simply put, a state represents one form which can be viewed by an authorized process participant, for example the aforementioned application reviewer. The attributes present in the state determine the form fields that are available for editing. Each object has an associated *micro process*, a flow graph which allows for the definition of an order in which the object can reach its various states, based on data-driven decisions. The object can only be in one state at any given point in time.

For example, a "Job Application" object, which is in the "Under Review" state, shows a generated form to the application reviewer. The generated form contains a field for the "Review Assessment" attribute. The state of the "Job Application" object changes to either "Accepted" or "Rejected", based on the value the reviewer assigns to the "Review Assessment" attribute. These state transitions, as well as the logic of one such form can be modeled in the object's micro process. In the flow graph of the micro process, each of the object's attributes is represented by a *micro step*. The arrangement of these micro steps within the individual states defines the form's logic. This is achieved by interconnecting the micro steps with transitions, thereby introducing an ordering. The transitions can also be used to model conditional branching.

Returning to the previous example of the "Application" object, the "Under Review" state of the object's micro process could be modeled as follows: The first micro step represents a string attribute "Internal Comment". The second micro step, representing another string attribute "Public Comment", is attached via a transition. Finally, the previously introduced "Review Assessment" attribute, an enumeration with the values "Good" and "Bad", is represented by the last micro step in the state. The "Review Assessment" micro step is attached to the "Public

Comment" micro step with another transition, creating the micro process model visible in Figure 1. The form which is generated for the "Under Review" state shows one form field for each of the attributes contained in the state. The fields are marked as *required* in the order induced by the micro process (cf. Figure 1).

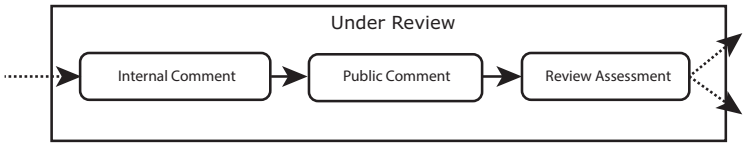

Fig. 1. Example of a State

As the execution is data-driven, the progress of the micro process advances to the next step as soon as a value for the current step is entered. As the PHILharmonicFlows Framework allows for flexible execution, similar to the case handling tool BPM—one [5], the process participant editing the form can input the values in any order he chooses. For instance, if the micro process reaches the "Review Assessment" micro step and a value is already present, the form does not need to mark the corresponding field as required. Instead, the execution continues with the evaluation of the current "Review Assessment" attribute value. An outgoing transition is chosen based on the result of this evaluation.

3 The PHILharmonicFlows Runtime Demonstration Tool

The PHILharmonicFlows runtime environment we present in this demonstration consists of a web service, hosting the PHILharmonicFlows runtime, as well as a client user interface, the Runtime Demonstration Tool (RDT)[1]. The RDT is implemented as a Windows Universal Application, allowing us to run the RDT on any device running Windows 8 or Windows 10. This means we can evaluate the feasibility of object-aware process management in various scenarios involving desktops, tablets, mobile phones, and even large tabletop computers, using a single unified codebase. Implementing the server as a web service will allow us to write web-based clients and native clients for other operating systems in the future. Client applications need only understand a simple WSDL containing the view model and small number of operations.

The current version of the client and server allow for the execution of micro processes (cf. Section 2). Micro processes describe the behavior of an object, i.e., which data attributes have to be provided when and which state changes to the object this entails. Figure 2 depicts the RDT in a split view showing the graph of one such micro process ①, as well as the generated form ② for the micro process' current state ③. The RDT always keeps the two displays, the

[1] A screencast is available at `https://vimeo.com/130551330`

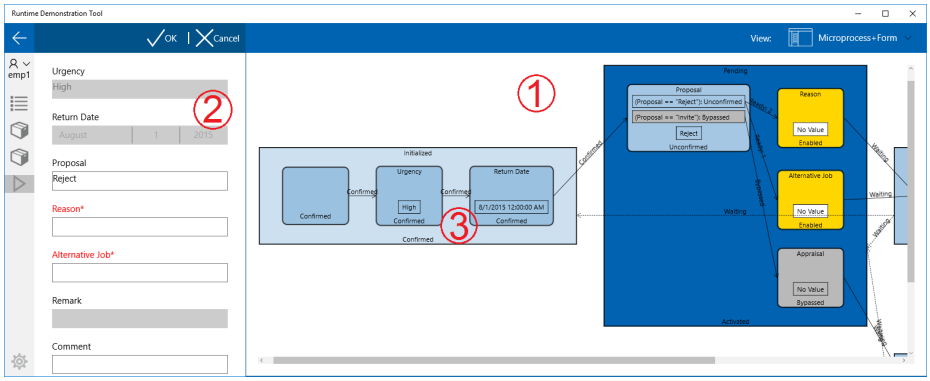

Fig. 2. Micro Process Execution View

form and the graph, in sync with the runtime server, where the actual micro process instance resides. When users of the RDT supply attribute values via the generated form or by editing micro steps directly in the graph, a web service is called, informing the runtime server of updated attribute values. The data-driven execution model of PHILharmonicFlows then evaluates the effects that the newly supplied or changed attribute value caused and sends change set deltas to the client, allowing it to update the displayed form.

In the example in Figure 2, supplying a value for the "Urgency" attribute causes the server to inform the client, that the "Return Date" attribute has to be marked as *required* next. Once both "Urgency" and "Return Date" are supplied, the "Review" object changes its state from "Initialized" to "Pending", causing it to show up in the worklist of any process participant who is authorized to view the "Pending" state. The process participant can then edit the form for the "Pending" state.

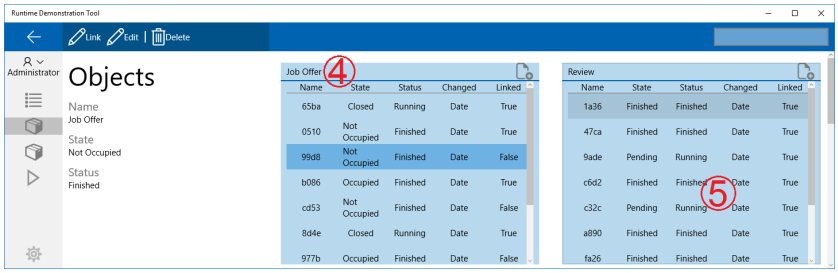

Fig. 3. Object Types and Instances Overview

The visualization of the micro process instance as a graph gives a quick overview on the progress of a micro process instance. The graph supports live

updates and changes immediately when other users concurrently edit the object's attribute values, thereby advancing execution of the micro process. The color coding of the micro process' states and steps according to their runtime *markings*, e.g. *Waiting* or *Activated*, helps us to explain and demonstrate the more complex points of the PHILharmonicFlows concept.

Figure 3 shows one of the menu pages of the RDT, specifically the "Objects" page. The "Objects" page gives authorized users an overview over all object types present in the current PHILharmonicFlows project. An object type is a build-time model of an object. The object type contains the definitions of the object's attributes, micro process and authorization settings. Each object type is represented by a tile ④ in Figure 3. Every tile contains a scrollable list of all the object instances ⑤ that exist of the object type. The list also shows some metadata determined by the current state of the object instance's micro process. Selecting an object type shows various meta information and allows users to create more instances of the object type. Selecting an already existing instance on the other hand allows users to edit the form generated for the instance's current state and view the live graph depicting the execution of the micro process instance (cf. Figure 2).

4 Conclusion

The demonstration presents parts of the PHILharmonicFlows runtime environment, consisting of a web service hosting the process server and a graphical user interface. Together, they allow for the execution of PHILharmonicFlows micro processes. The user interface enables interaction with running process instances via generated user forms, worklists, and graph models. While the current version only supports micro processes, we are working on adding support for *macro processes*, which enable interaction between multiple objects' micro processes, allowing for far more complex processes to be executed.

References

1. Künzle, V.: Object-Aware Process Management. Ph.D. thesis, University of Ulm (2013)
2. Künzle, V., Reichert, M.: A Modeling Paradigm for Integrating Processes and Data at the Micro Level. In: Enterprise, Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing, vol. 81, pp. 201–215 (2011)
3. Künzle, V., Reichert, M.: PHILharmonicFlows - Towards a Framework for Object-Aware Process Management. Journal of Software Maintenance and Evolution: Research and Practice 23(4), 205–244 (2011)
4. Künzle, V., Weber, B., Reichert, M.: Object-Aware Business Processes: Fundamental Requirements and their Support in Existing Approaches. Int'l Journal of Information System Modeling and Design (IJISMD) 2(2), 19–46 (2011)
5. Lexmark Enterprise Sofware: Lexmark Case Management Tool (2015), `http://www.perceptivesoftware.com/products/perceptive-process/case-management.html`, last accessed on 2015/06/12