A Modeling Tool for PHILharmonicFlows Objects and Lifecycle Processes

Sebastian Steinau, Kevin Andrews, and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Germany
{firstname.lastname}@uni-ulm.de

Abstract. As opposed 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 implementing data-centric, artifact-centric or object-aware approaches. This demo paper presents the implementation of a modeling environment for the object-aware approach to business process management. Our implementation is based on the PHILharmonicFlows framework, which allows for the definition and execution of business object models. The current implementation of the modeling environment supports the modeling of objects, their attributes as well as relations to other objects. Furthermore, it allows modeling object lifecycle processes, which define the runtime behavior of the various objects. Finally, the modeling environment features a simple app design, reducing complexity for process modelers while still supporting all features of PHILharmonicFlows.

1 Introduction

Increasing flexibility in process management and execution has been broadly discussed in literature in recent years [5]. While most research has focused on the flexibility of traditional activity-centric process execution, a number of entirely new process management paradigms have emerged [2,3,4]. Their focus lies more on the data or cases involved in a process and less on the strict ordering of activities in a single process model. One of these paradigms is *object-aware process management*, which takes the business objects present in real-world processes as the basis for composing an executable process model, which, itself, comprises models for each object, as well as its lifecycle during process execution and its data. By creating processes revolving around largely independent objects, object-aware processes, in many cases, allow for greater flexibility compared to activity-centric processes [4].

As the concept of object-aware process management is sophisticated and requires a shift in perspective and thinking, when creating the modeling environment for object-aware processes, our main goal was to reduce complexity for users to a minimum. We consider this a prerequisite for future studies of the paradigm, as participants should be able to collaboratively model object-aware

processes without having vast amounts of prior knowledge. We achieve this goal by using a simplistic app design as well as real-time verification. Real-time verification helps reduce the difficulty of modeling as it can guide users, preventing them from modeling defective processes. Furthermore, the modeling tool should be extensible, to support future conceptual additions, such as schema evolution and process variants.

The tool we present in this paper[1] uses the *PHILharmonicFlows* implementation of object-aware process management at its core, thereby allowing for the graphical creation of process models that run on the existing PHILharmonicFlows runtime environment [1]. As we are currently in the process of completely restructuring the architecture of the PHILharmonicFlows engine to create a highly scalable distributed process execution architecture, an additional challenge was to ensure that the modeling environment as well as its collaborative modeling and real-time verification capabilities are compatible with both the current client-server architecture, and the new distributed architecture.

2 Modeling Objects and Relations

The object-aware approach to process modeling utilized by the modeling tool can be illustrated using a comprehensive scenario (cf. Example 1).

Example 1. Simplified recruitment process
In the context of recruitment, applicants may apply for job offers. The overall process goal is to determine the applicant best suited for the job. To evaluate an application, reviews need to be performed. Depending on the concerned department, the number of reviews may differ. Department employees write the reviews and either reject the applicants or suggest inviting them for an interview. In the meantime, more applications may have arrived, for which further reviews are required. This allows for the evaluation of different applications in parallel. If the majority of reviews are in favor of an application, the applicant is invited for an interview, in which he may be hired or rejected. Finally, when one applicant is hired, all other applicants shall be rejected.

At design time, PHILharmonicFlows uses various *types*, which serve as templates to create *instances* at runtime. As a first step to modeling, it is necessary to identify the *object types* involved. These are directly derived from the business objects and persons involved in the real-world process. In Example 1, object types include *Job Offer*, *Application*, *Review*, and *Interview*. *Employee* is a *user type*, a special case of an object type representing a person. Each object type has a set of *attribute types*, describing the data of the corresponding objects. The attribute types are primitives, as they are required for the generation of individual form fields at runtime. An example attribute type could be *Comment*, a *String* defined in the *Review* object type. Finally, each object type may be connected other object types with *relation types*, allowing their logical association at runtime. Together, object types and relation types form the *data model*.

[1] A screencast is available at `https://vimeo.com/222018637`

Figure 1 illustrates the creation of the data model related to the example. The data model comprises object types *Employee*, *Job Offer*, *Application*, *Review* and *Interview* and is shown in main area ①. On the left ②, all modeling elements available in the given context are shown. The data model is expanded by dragging a modeling element and dropping it onto the main area. The edges representing the relation types can be created by connecting object types with a simple mouse gesture. The layouting of the data model graph is done automatically, i.e., users need not concern themselves with layouting.

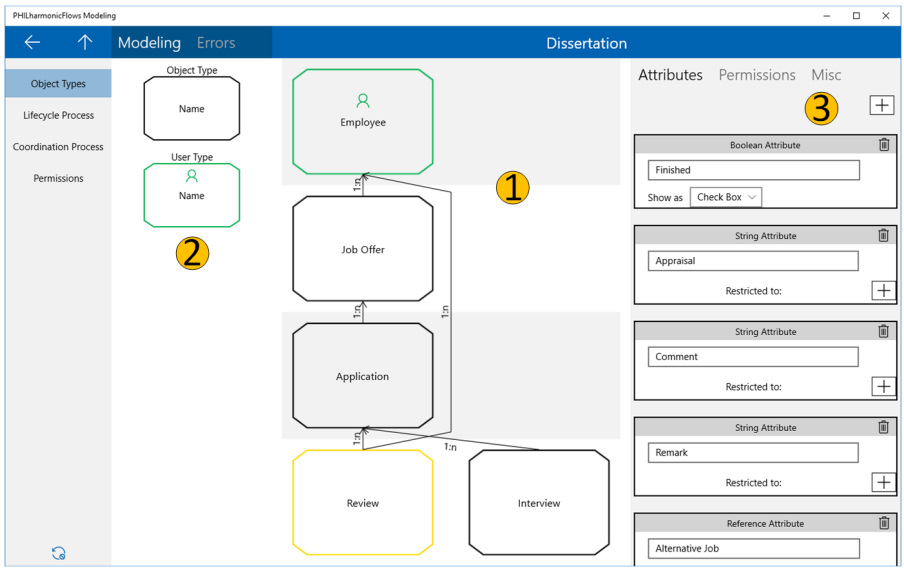

Fig. 1. Object and relation type modeling view

In the right-hand sidebar ③, object types and relation types may be configured after selecting them in the main area. Figure 1 shows the attributes of the *Review* object type in the attributes tab. The existing attributes can be altered or deleted and new attributes may be added.

3 Modeling Lifecycle Processes

Each object and user type is required to have a *lifecycle process* (cf. Fig. 2). As lifecycle processes in object-aware process management are data-driven, they describe which attributes need to have assigned values before an object may change its *state*. States ④, in turn, become necessary to coordinate the processing of an object with the processing of others based on their current states. Within states, *steps* ⑤ are used to require an attribute value at runtime. Each step is assigned to an attribute type and connected to other steps by *transitions* ⑥, thereby determining the order in which values for the various attributes are required at runtime, i.e., the logic of the corresponding electronic form.

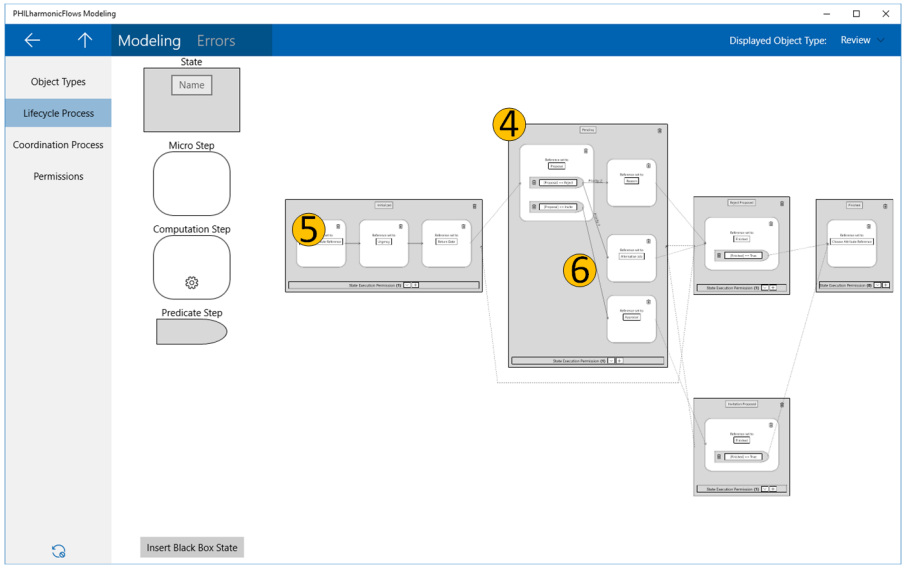

Fig. 2. Lifecycle modeling view

The lifecycle modeling view (cf. Fig. 2) is arranged analogously to the object and relation modeling view (cf. Fig. 1), thereby ensuring a familiar user interface for all parts of the modeling tool. The right-hand sidebar is not displayed, but opens upon selecting an element in the main area, allowing for its configuration.

Overall, the modeling tool aims at providing support for the complex modeling concept of PHILharmonicFlows in as simple a fashion as possible. All modeling actions are immediately synchronized to the PHILharmonicFlows server for immediate verification and to allow for collaborative modeling. Immediate verification and feedback helps users to not model incorrect processes.

4 Model Verification

Model verification is necessary to prevent incorrect process models from being deployed to the PHILharmonicFlows runtime environment. The verification is divided into different verification components, each checking one aspect of a model, e.g., graph acyclicity or steps without outgoing transitions. The separation of verification into individual components becomes necessary, as verification of the entire data model after every modeling action is not feasible. Immediate verification, therefore, only uses a subset of all available verification components. These verification components provide meaningful error messages, which are necessary for guiding the user through the complexities of PHILharmonicFlows process modeling. Figure 3 shows an error message for multiple start steps in a state ⑦. Furthermore, upon selecting the error message, the involved process elements in the main area are marked to help the user pinpoint the error location ⑧. Prior to deployment, a full verification with all components is required.

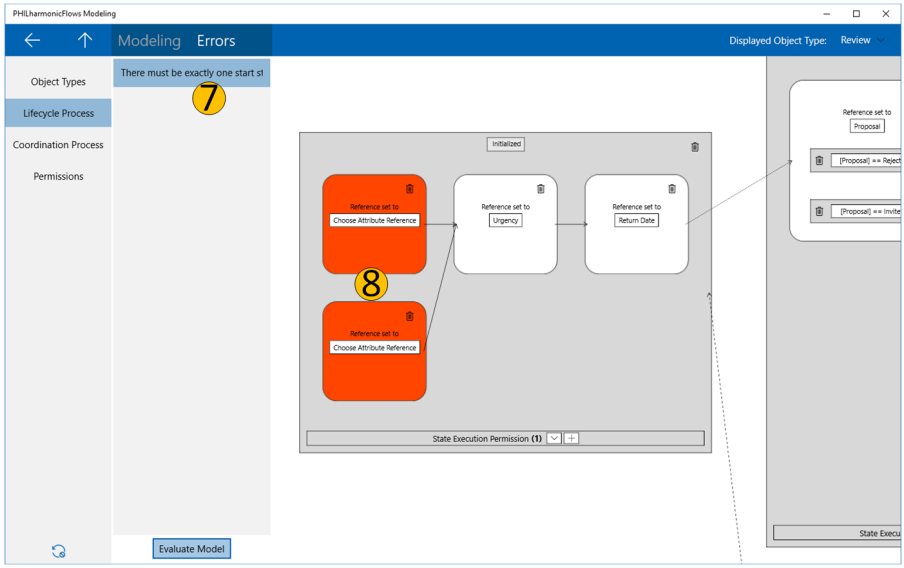

Fig. 3. Error message and colored modeling elements

5 Summary and Outlook

The modeling tool presented in this paper is a first step in our ongoing effort to make modeling object-aware processes feasible without needing an in-depth knowledge of the entire object-aware concept. Furthermore, the tool offers a simple user interface with real-time model verification feedback while modeling. Without such tooling support, modeling an object-aware process can be complicated, as changes may impact entirely other parts of the data model. Finally, the modeling environment is extensible with further capabilities, as necessary for our future research into schema evolution and variability of object-aware processes.

References

1. Andrews, K., Steinau, S., Reichert, M.: A Runtime Environment for Object-aware Processes. In: BPM Demo Session (BPMD). CEUR Workshop Proceedings (2015)
2. Haddar, N., Tmar, M., Gargouri, F.: Opus Framework: A Proof-of-Concept Implementation. In: IEEE/ACIS 14th Int'l Conf. Computer and Information Science (ICIS). pp. 639–641 (2015)
3. Heath, T.F., Boaz, D., Gupta, M., Vaculín, R., Sun, Y., Hull, R., Limonad, L.: Barcelona: A Design and Runtime Environment for Declarative Artifact-centric BPM. In: 11th Int'l Conf. on Service-Oriented Comp. (ICSOC). pp. 705–709 (2013)
4. 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)
5. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies (2012)