Oryx – Sharing Conceptual Models on the Web

Gero Decker, Hagen Overdick, and Mathias Weske

Hasso-Plattner-Institute, University of Potsdam, Germany
(gero.decker,hagen.overdick,weske)@hpi.uni-potsdam.de

Abstract. In this paper we present the architectural blueprint and a prototypical implementation of Oryx, a visual environment that facilitates zero-installation for creating, sharing, and documenting conceptual models.

1 Introduction

In recent years, the complexity of software systems has risen sharply, so that the role of conceptual modeling is more important than ever. To capture this complexity, different groups of individuals are now involved in modeling different aspects of the system, rather than a few people modeling internals of a software system. These different groups of persons concentrate their modeling effort on different aspects of the system and use different modeling techniques, for instance UML structure diagrams and the Business Process Modeling Notation.

Conceptual models are developed in a collaborative way, models are shared, discussed, reviewed, and finally agreed upon. In process modeling, for instance, organizational borders might be crossed, so that experts from different companies discuss their business processes and how these interact.

These characteristics of conceptual modeling impose requirements on modeling tools. In this demonstration paper we report on Oryx, an extensible modeling framework that makes use of Web 2.0 technologies [2]. The Oryx framework can be tailored to support different modeling languages; it does not require software installation on the client side, just an off-the-shelf web browser.

In Oryx, each model artifact is identified by a URL, so that models can be shared by passing references, rather than by exchanging model documents in email attachments. Since models are created using a browser and models are just "a bookmark away", contribution and sharing of conceptual models is eased.

The rest of the paper is structured as follows. Section 2 highlights the most important requirements for the use case scenarios addressed. Section 3 outlines how these requirements are addressed in the Oryx framework. Section 4 focuses on the BPMN support that is realized in Oryx. Finally, a conclusion is given in Section 5.

2 Oryx Use Cases

Support for Multiple Languages Modeling takes a central role in various disciplines of computer science, including model driven development, database design,

software architectures, and business process management [3]. Models are domain-specific abstractions used for documentation and exchange of ideas, decisions, and operation guidelines, but also as blueprint for system design and development. Even within one individual domain there is a wide range of notations in use, i.e. there is no common metamodel, but domain-specific notations are used. Therefore, support for multiple modeling languages is a key requirement for Oryx.

Meta-Information and Feature Extensions Much effort is spent on using models as a constructor of reality, i.e. using models to build new systems. An obvious example is a process model, which is instantiated by a workflow engine resulting in a system behavior according to the encoded process specification. Execution environments typically require large sets of meta-information augmenting the model elements. Examples are execution probabilities on edges, which are used in simulation, monitoring, and visualization; in case Web service are used to execute processes, technical configurations need to be represented. To do so, there must be a strategy how to extend models and how to provide plugins operating on these extensions, resulting in the second key requirement.

Data Portability Models are relevant to many stakeholders, e.g. system architects, developers, customers, and end-users. Their individual use-cases for models varies to a large extent. Consequently, stakeholders will use different tools when working with a model, e.g. the software architect needs an editor, the analyst uses a validator, the developer wants input to a code-generator, and end-users just want a good viewer to get an understanding of the model. Some of these scenarios are not even browser-based, yet need to be supported. Data portability means the opportunity to use well documented model data formats that can be used by different tools.

3 Oryx Architecture

Figure 1 depicts the Oryx architecture. While the current release requires a specific Oryx backend, in theory any location on the Web will do. Oryx itself is just a set of Javascript routines loaded into a modern web browser as part of a single document describing the whole model. Models are represented in RDF format. The Oryx core provides generic handling of nodes and edges; how to create, read, and update them; as well as an infrastructure for stencil sets and plugins.

Language Support via Stencil Sets Stencil sets drive the Oryx core, as they provide explicit typing, connection rules, visual appearance, and other features differentiating a model editor from a generic drawing tool. Hence, a stored Oryx model is structure first, directly based on the loaded stencil sets, visual diagram second.

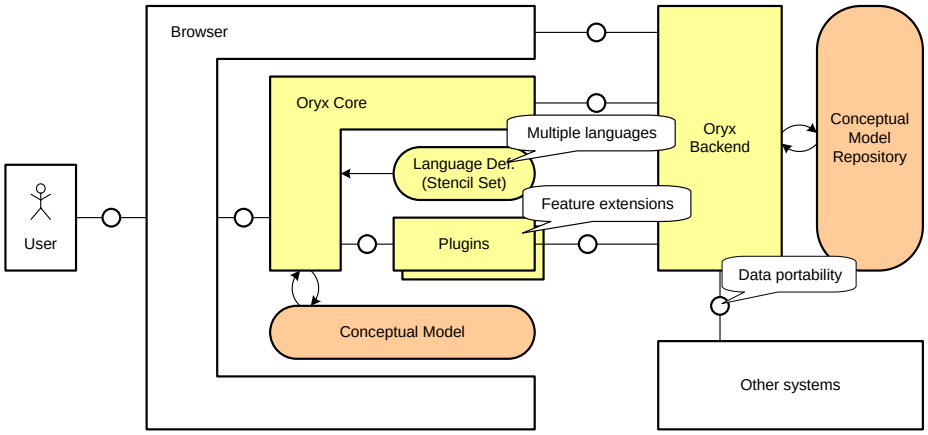

Fig. 1. Oryx architecture

Feature Extensions via Plugins Plugins allow for both generic as well as notation specific extensions. E.g. even element selection and cut & paste are plugin features, as they are not needed for an Oryx viewer. More advanced plugins realized allow for complex model checking beyond the powers of the stencil set language.

Data Portability beyond Oryx The Oryx core, with the help of stencil sets and plugins, allows users to create, edit, and view visual models within a browser. Currently, Oryx does so by self-modifying the loaded page and sending it back to the server in whole.

Still, any model element created by Oryx is addressable via a URI. The returned representation will turn into an Oryx editor when Javascript is turned on and can be transformed into RDF via XSLT.

4 Oryx for BPMN

Any tool can claim to be open, extensible, and hence the one-stop solution in all scenarios. To support our claim, a strong initial focus was put on supporting the Business Process Modeling Notation (BPMN) [1]. BPMN suits well as an example for several reasons: First, it is an important notation in the business process community that the authors belong to. Second, BPMN is designed to be easily understood, even by people outside the business process modeling domain. Moreover, while there are efforts, there is still no generally accepted serialization format for BPMN, making it easy to discuss requirements and challenges without being tempted to use an existing format. BPMN requires many features beyond a simple drawing tool, e.g. element typing, containment rules, and type specific layouting. Oryx today has full support for BPMN 1.0. Existing features build on top, both by our research group and colleagues from academia, include BPMN to Petri net mapping, BPMN to BPEL transformation, XPDL serialization for BPMN, as well as deadlock analysis for BPMN.

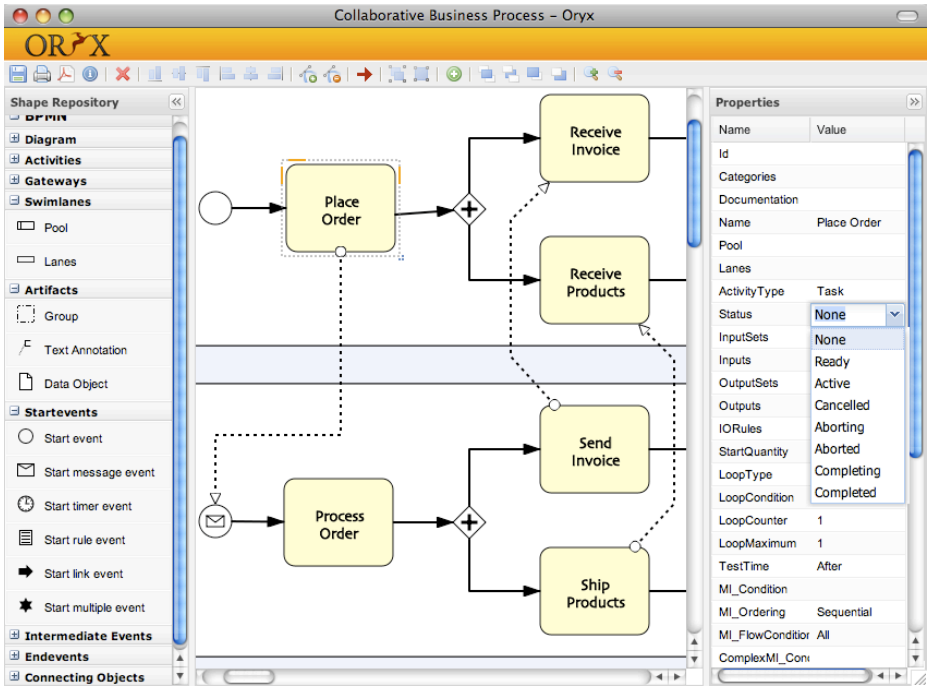

Fig. 2. Oryx for BPMN: Modeling by drag and drop of notational elements shown on the left hand side. Right hand side shows all BPMN 1.0 attributes of a selected model element, here the Place Order activity.

5 Conclusion

This paper has presented Oryx, an extensible framework for conceptual modeling on the web. In research collaborations, new notations and features have been added easily. Oryx is an open source project under MIT license. The Oryx homepage can be found at `http://bpt.hpi.uni-potsdam.de/Oryx/`. Interested parties are welcome to use Oryx and to contribute to it.

Acknowledgements: The authors thank the Oryx team at HPI for their work, including Martin Czuchra and Ole Eckermann. The work of student projects that contributed to earlier versions of the system is acknowledged as well.

References

1. Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification. Technical report, Object Management Group (OMG), February 2006.
2. G. Vossen and S. Hagemann. *Unleashing Web 2.0: From Concepts to Creativity.* Morgan Kaufmann, 2007.
3. M. Weske. *Business Process Management.* Springer, 2007.