Direwolf Model Academy: An Extensible Collaborative Modeling Framework on the Web

István Koren, Ralf Klamma, Matthias Jarke[1]

Abstract: Conceptual modeling in Industry 4.0 scenarios enables orchestrating production processes and planning data flows. Since diverse stakeholder groups are involved, collaboration features are particularly important. Common web-based tools are available; however, they focus on either modeling or collaboration. Based on our experiences with these two aspects, we present Direwolf Model Academy, a metamodel framework for creating feature-rich modeling environments on the web. It is based on modern standards like SVG and Web Components; it uses object-oriented programming principles enabled by the latest generation of JavaScript. We already employ tool instances in the areas of user interface generation from service description languages as well as conceptual modeling with iStar 2.0. In this article, we discuss the latter implementation and present the underlying technological foundation. Our framework is available open source on `https://github.com/direwolf` where we welcome contributions.

Keywords: Collaborative Modeling; Metamodeling; Web Engineering

1 Introduction

Conceptual modeling is a highly social activity, involving several expert opinions [JJM09]. This is particularly valid in the realm of Industry 4.0, where interdisciplinary teams work together to create abstractions of production processes and work plans. Digitized industrial artifacts can integrate new functionalities and collect data at any time, even after physical production. Thus, emerging alliance-driven data platforms require business modeling as flexible co-creational activity. To this end, domain-specific modeling languages contribute to transforming complex resource architectures into policies, and ultimately real-world systems through code generation. Thereby, agile principles need to be respected to enable fast prototyping and cater for changing requirements. Co-location of modelers is time-consuming and expensive; to support them, collaborative, light-weight and accessible tools are necessary. There are several existing tools like MetaEdit+, and ADOxx that partially address the requirements above [De15]. Comparing the available tools, one can observe that most tools lack the web collaboration functionality and put more emphasis on the implementation of application-specific features like code generation in software engineering or simulation of business process models. However, these tools require local installation and maintenance support. Browser-based solutions provide an ideal environment due to

[1] RWTH Aachen University, Lehrstuhl Informatik 5, Ahornstrasse 55, 52074 Aachen, Germany, {koren,klamma, jarke}@dbis.rwth-aachen.de

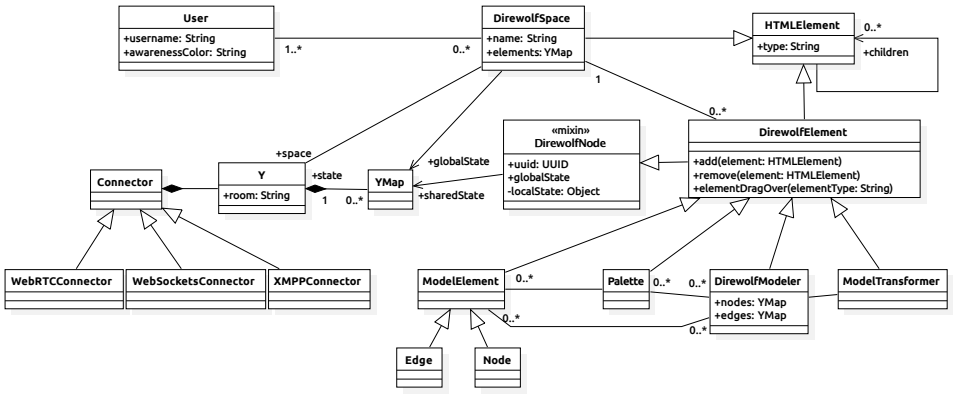

Fig. 1: Direwolf Model Academy Architecture as UML Class Diagram

the universal availability across devices. On the web, near real-time collaborative features are ubiquitous in end user software like Google Docs and Microsoft Office. However, approaches connecting agile modeling with near real-time collaboration are missing. To this end, in this article we present *Direwolf Model Academy*, a JavaScript framework for creating synchronous collaborative web applications for visual modeling. Our metamodel creation approach is rooted in web and software engineering research on involving end users through *infrastructuring* [PS06]. In the next section, we present our software architecture that has a synchronized data structure in its core, on which model transformers can work to generate model views on-the-fly. The framework provides ready-made UI views for palettes and property browsers. In Section 3 we then demonstrate a concrete instance, the Direwolf iStar 2.0 editor for modeling strategic dependency diagrams in the iStar 2.0 notation. We conclude the paper in Section 4 with an outlook on future work and an invitation to submit issues on our GitHub repository.

2 Object-Oriented Metamodel Design

Our main requirement was to create a highly modular library for creating universal modeling applications on the web. It uses web standards like HTML5, CSS3 and SVG. The framework needs to be versatile enough to support various edge- and node-based metamodels and their view-based transformations for diverse application cases. The challenge is to create a software architecture that is collaborative in near real-time, i.e., below a human-perceivable reaction time. The resulting Direwolf Model Academy is flexible enough for graph-based representations of modeled artifacts and their connections. It comes with extension points to create bidirectional model-to-model transformations. Figure 1 shows a simplified UML class diagram of our application. On the left, all classes responsible for collaboration are shown. The central *DirewolfSpace* has access to a synchronized global data store; it is synchronized across application instances running on distinct browsers. Conceptually, it is an HTML

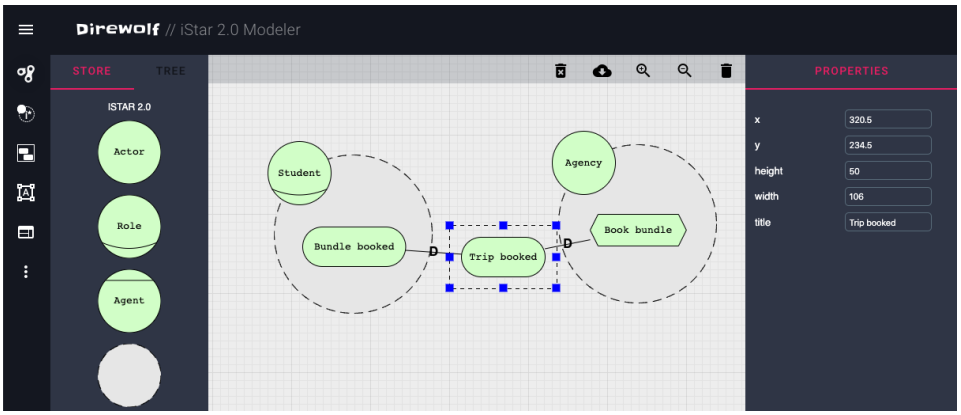

Fig. 2: Screenshot of Direwolf iStar 2.0 Modeler

element, as are all child nodes. The abstract *DirewolfNode* provides all elements with shared data structures. For instance, *DirewolfModeler*, the graphical engine that enables drawing nodes and edges, holds its data in the synchronized maps *nodes* and *edges*. Simultaneous users can each have different views on the underlying data. *ModelTransformer* instances similarly have access to the shared data store, however they may transform the data to different representations through code generation; an example is the generation of data access policies based on a data access model.

For the visual display, we leverage that the class of graph-based visual modeling languages of nodes and edges are relatively simple in their graphic expression and can be composed of vector-based graphical primitives. They are translated to an inheritance-based object model of *Node* and *Edge* instances. On every level, properties are synchronized. Thus, To resize a node, the visual model editor needs to change the width property of the top-most element. Then our framework sends an event to both the base class and the derived elements. The change is propagated down the hierarchy through the Direwolf shared data store.

3 Proof of Concept

The Direwolf Model Academy framework and its modeling application instances are implemented using JavaScript. The graphical components of a model are implemented as Scalable Vector Graphics (SVG) elements. Thus, the resulting file can be directly exported to vector-based graphics program like Microsoft Visio and draw.io. Synchronization is done via the Yjs open source library (`https://yjs.dev`). The peer-to-peer nature of Yjs ensures that no centralized entity is responsible for merging conflicts, leading to better scalability characteristics. Models can be worked on while the system is not connected to the Internet; the data structures are synchronized with peers when reconnecting.

The components visible in the user interface of Figure 2 are explained in the following. A **Modeler** is a graphical editor that displays the graph- and edge-based composition of a model. All modelers create a tree-based data structure. It is synchronized using a publish-subscribe pattern; upon changes, an event is sent to the modeler which then updates the graphical representation. **Palette** elements (on the left) provide a list of nodes and edges. They may be specified at design-time, or dynamically generated at run-time. From here, model elements can be moved to the modeler via drag-and-drop. **Property Browsers** (on the right) are form-based representations of a selected model element's properties. Technically, the browser gets access to the shared data store of the model's class. Once changed, the publish-subscribe system notifies the model element's visual instance in the modeler view, to change properties, e.g. the size. **Transformers** use the same event API as modelers, however they transform the model either to a different metamodel instance, or to an application-specific representation.

4 Conclusion and Future Work

In this article we presented Direwolf Model Academy, a standards-based metamodel framework whose instances are synchronized across browsers in near-realtime. Besides modeling data access goals with iStar 2.0, we already employed it for modeling UIs for REST-based services; here, the palette elements are automatically generated from a service description. Due to the extension points of its software architecture, there are manifold opportunities for extending the framework. Currently, we are working on user authentication and awareness features. Beyond, we plan to model Industry 4.0 data streams in near real-time with code generation for edge-oriented interfaces. On our GitHub repository (`https://github.com/direwolf`), we are open for inquiries concerning further conceptual models and other collaborations. A demo is deployed on `https://direwolf.rocks/spaces`.

Acknowledgment. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

Bibliography

[De15] Derntl, Michael; Nicolaescu, Petru; Erdtmann, Stephan; Klamma, Ralf; Jarke, Matthias: Near Real-Time Collaborative Conceptual Modeling on the Web. In (Johannesson, Paul et al., ed.): 34th International Conference on Conceptual Modeling (ER 2015). volume 9381 of Lecture Notes in Computer Science, Springer International Publishing, Cham, Switzerland, pp. 344–357, 2015.

[JJM09] Jeusfeld, Manfred A.; Jarke, Matthias; Mylopoulos, John, eds. Metamodeling for Method Engineering. MIT Press, 2009.

[PS06] Pipek, Volkmar; Syrjänen, Anna-Liisa: Infrastructuring as Capturing In-Situ Design. In: 7th Mediterranean Conference on Information Systems. 2006.