How Web 2.0 can leverage Model Engineering in Practice

M. Wimmer[1], A. Schauerhuber[1]*, M. Strommer[1],
J. Flandorfer[1], and W. Schwinger[2]

[1] Institute of Software Technology and Interactive Systems
Vienna University of Technology, Austria
{wimmer|schauerhuber|strommer|flandorfer}@isis.tuwien.ac.at
[2] Department of Telecooperation, Johannes Kepler University Linz, Austria
wieland.schwinger@jku.ac.at

Abstract. Today's online model repositories offer to download and view the textual specifications of e.g. metamodels and models in the browser. For users, in order to efficiently search a model repository, a graphical visualization of the stored models is desirable. First attempts that automatically generate class diagrams as e.g. PNG bitmaps, however, do not scale for large models and fail to present all information. In this paper, we present our Web 2.0 MetaModelbrowser, a model visualization service which provides an Ajax-based tree-viewer for efficiently browsing Ecore-based metamodels and their models. As a main contribution of this work the MetaModelbrowser is complementary to existing model repositories in that its visualization service can be integrated into them. The MetaModelbrowser, furthermore, allows zooming in and out of the details of arbitrarily sized models as necessary.

Key words: model repositories, model visualization, Web 2.0, Ajax

1 State-of-the-art in Model Repositories

With the rise of model-driven development, model repositories are intended to facilitate research in model engineering and consequently in model-driven web engineering. Model repositories are central places where all kinds of modeling artifacts (e.g., metametamodels, metamodels, models, and possibly transformation models) are stored and coordinated. They can serve as a platform for making available the specification of metamodels to others and for exchanging models, as well as a resource for teaching/learning materials.

While some initiatives for building model repositories have been started (e.g., zoomm.org, www.kermeta.org/mrep), the currently most comprehensive repository of models is hosted within the open source ATLAS MegaModel Management (AM3) subproject [1] of the Eclipse Generative Modeling Technology (GMT)

* This research has been partly funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

project. The artifacts present in this model repository, furthermore, are organized into sets of models of similar nature called zoos, e.g. a zoo for metamodels and a zoo for transformations [2].

Generally, today's online model repositories offer either to view the models' textual specifications in standard Web browsers or their download for graphically viewing them offline with appropriate tools, e.g. the Eclipse Modeling Framework's (EMF) *Sample Ecore Model Editor* [3]. Neither possibility is satisfactory: On one hand the textual representation of models is hardly readable and understandable for humans. On the other hand, downloading a model for an offline graphical visualization first, requires the appropriate tools installed, second, has to be done for each model as well as detaches the downloaded models from the rest of the zoo, and third, is time-consuming.

For users, in order to efficiently search a model repository, a graphical visualization of the stored models is desirable. First attempts, such as the AM3 Atlantic Raster Zoo, that automatically generate class diagrams as PNG bitmaps, however, do not scale for large models. More specifically, the readability of these pictures is hampered by some modeling elements (e.g., associations and roles) hiding others. Furthermore, users are always confronted with the whole picture and cannot show or hide parts of the model as necessary. Finally, the class diagram representation fails to present all information of each model element.

2 Web 2.0 to the Rescue

As a solution to the above presented problems, we propose our Web 2.0 MetaModelbrowser (MMB) as an alternative way of visualizing models within online model repositories. This MMB offers an Ajax-based tree-viewer for efficiently browsing Ecore-based metamodels and their models. In this respect, our goal is that users of the service get an idea of the models' structure as quick as possible. A tree-viewer, enables such efficient browsing by allowing to zoom in and out of the details of arbitrarily sized models as necessary.

Thus, our idea is to reproduce the user interface of EMF's Sample Ecore Model Editor (cf. Figure 1 (a)) for the Web. More specifically, parts of EMF's architecture and plug-ins are reused and the JFace/SWT-based user interface of the desktop application is replaced (cf. Section 3). Figure 1 (b) provides a screenshot of the MMB visualizing the SimpleUML metamodel (i.e. a simplified version of the UML class diagram metamodel), which is available at the AM3 AtlantEcore Zoo.

It has to be noted that in this work we seize two aspects of the Web 2.0. First, seen from the social network aspect, model repositories in fact are Web 2.0 applications that at the same time boost and are supported by the model engineering community. Second, seen from the technology aspect, the use of Ajax - often presented as the Web 2.0 enabling technology - allows for a desktop like working experience avoiding full page reloads and allowing user-transparent page updates.
Our contributions are briefly summarized in the following:

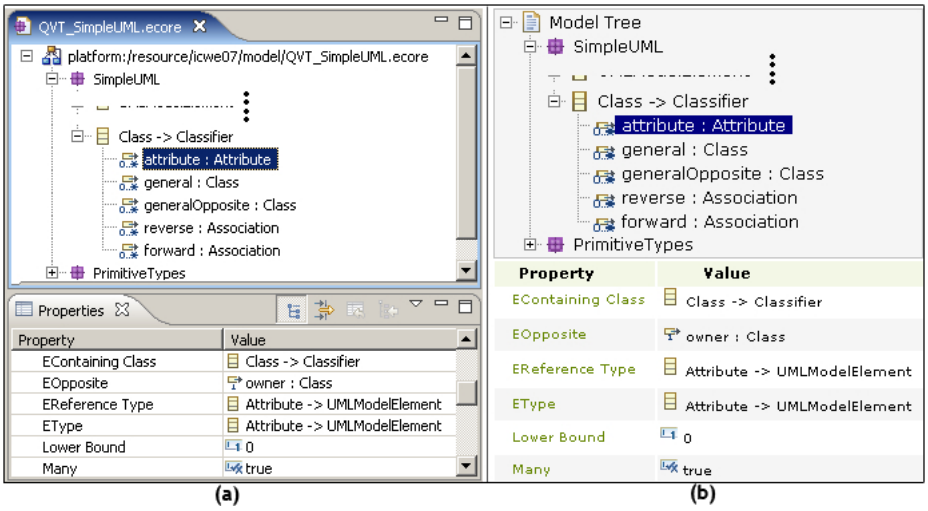

Fig. 1. SimpleUML metamodel in (a) Eclipse and (b) MMB

Extension for Existing Model Repositories. The MMB is complementary to existing model repositories in that its visualization service can be integrated into them. This is done by including parameterized links to the MMB into one's own model repository website as explained in Section 3.

Usability. In contrast to automatically generated class diagram pictures, a tree-viewer, allows to zoom in and out of the details of arbitrarily sized models as necessary. Moreover, the MMB provides information of the models that cannot be captured in class diagrams in a separate properties table for each model element (cf. Figure 1 (b)).

Saving of Time. There is no need to download the models, as they can be visualized in the browser. Furthermore, the asynchronous communication through Ajax allows to request only those parts of a model which the user is interested in.

Familiar Environment. The MMB's user interface is heavily based on the Eclipse Sample Ecore Model Editor and thus, represents a familiar environment to EMF users (cf. Figure 1).

Reuse. We have built the MMB upon existing EMF plug-ins. Beside being able to browse Ecore-based metamodels, this allows automatically generating the necessary artifacts for Web 2.0 *Model*browsers. This means that users can also browse models conforming to the metamodels (cf. Section 3.2).

Comparability of Models. The visual representation of models fosters their comparability, e.g., metamodels representing the same language such as UML 2.0 metamodel specifications from different authors.

Intellectual Property. If necessary, an additional feature of our MMB allows uploading the models to the server. The models can then be browsed via an ID but the source file is not given away.

3 The Web 2.0 MetaModelbrowser

In the following sections we outline the implementation of our MMB. Its visualization service and examples can be accessed at www.metamodelbrowser.org. In order to use our MMB for visualizing models the inclusion of the following parameterized URL in one's own model repository website is necessary:
www.metamodelbrowser.org/BrowseTreeServlet?url=⟨URL_OF_MODEL⟩

3.1 Architecture of the MetaModelbrowser

As a basis for our implementation task we have chosen to make use of the EMF framework whose main purpose is to allow for building tree-based model editors. EMF therefore provides for its own metamodeling language called Ecore, which can be seen as equivalent to the OMG's Essential MOF [4]. As is shown in Figure 2 (a), Ecore-based models can then be fed into the EMF code generator to automatically produce all components for a fully functional model editor. Those components are in fact three distinct Eclipse plug-ins, i.e., one representing the model, one that acts as a controller between the model and the viewer, as well as one that represents the user interface and that is integrated within the Eclipse workbench.

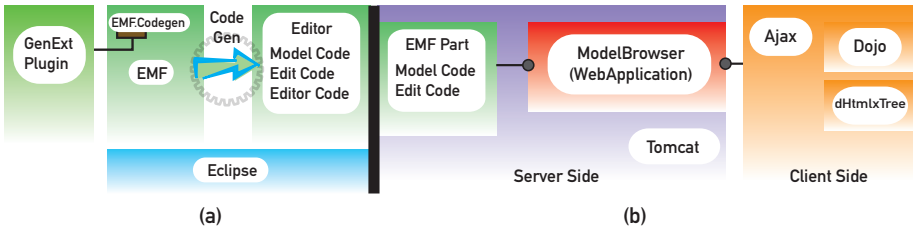

Fig. 2. EMF components overview (a) and MMB architecture (b)

The MMB architecture, depicted in Figure 2 (b), is based on the Tomcat server infrastructure and thus relies heavily on the Java Servlet and JSP technologies. Concerning EMF we completely reuse the model and edit code from EMF's built-in Sample Ecore Model Editor as external libraries. To replace the JFace/SWT editor code we have come up with a browser-based viewer, that can be populated with content of any arbitrary Ecore-based model. This viewer is realized with the Ajax frameworks dHtmlxTree [5] and Dojo [6] as shown in Figure 2 (b). dHtmlxTree serves as a Widget for displaying the model elements in a tree. To display the properties of model elements in a table we used Dojo.

3.2 Browsing Models

In the above section we described how to browse metamodels by reusing existing EMF components and integrating them within the web application. More chal-

lenging, however, is the ability to browse also models and the design decision we have had to make at this point in time. To be able to browse models one needs to install our GenExtPlug-in (cf. Figure 2 (a)), that uses an extension point of EMF.Codegen in order to apply some changes to the code generation process. After installing the plug-in the user needs to generate model and edit code and export these two artifacts as plug-ins. These plug-ins in turn have to be uploaded into the web application and can be used as soon as they reside on the server. Our MMB then automatically determines what Modelbrowser to instantiate on the basis of the model to be visualized. For demonstration purposes we have used our GenExtPlug-in to generate the code for the SimpleUML Modelbrowser, that is available at our project site.

4 Future Work

The presented work reveals two major directions for future work.

What is an intuitive visualization of models and is the tree-based view an appropriate one? First of all, we believe that an intuitive visualization should consist of several views on the model, depending on the user's interests. The tree-based view is focused on depicting the Ecore metamodeling language's structural relationships between packages, their classes and in turn their features. Nevertheless, a class diagram can provide a better visualization for e.g., inheritance and containment relationships, under certain conditions. In this respect, a static class diagram picture is not enough. The user has to be able to influence the way and what parts of a model are to be visualized like in existing desktop modeling environments, e.g., temporarily hiding of any model element. Ideally, the class diagram view and the tree-view are combined and the latter serves as an outline of the model, e.g. in a sidebar. These ideas directly lead to the second direction for future work.

Is it possible to realize a model editor as a web application? Our MMB allows read-only access to models but could be extended for editing functionality. The introduction of such functionality, however, comes along with other important issues such as versioning and concurrency which have already been researched in the database community.

References

1. ATLAS MegaModel Management, official site: http://www.eclipse.org/gmt/am3
2. Allilaire, F., Bézivin, J., Bruneliére, H., and Jouault, F.: Global Model Management in Eclipse GMT/AM3. Eclipse Technology eXchange workshop at ECOOP, 2006
3. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.: Eclipse Modeling Framework. Addison Wesley, 2003
4. OMG: Meta Object Facility (MOF) 2.0 Core Specification, 2003
5. dhtmlxTree - AJAX powered DHTML JavaScript Tree component with rich API, official site: http://scbr.com/docs/products/dhtmlxTree/
6. Dojo - The Javascript Toolkit, official site: http://dojotoolkit.org/