How Web 2.0 can leverage Model Engineering in Practice

Manuel Wimmer, Andrea Schauerhuber, Michael Strommer,
Jürgen Flandorfer and Gerti Kappel
Business Informatics Group
Institute for Software and Interactive Systems
Vienna University of Technology, Austria
{wimmer|schauerhauber|strommer|flandorfer|kappel}@big.tuwien.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 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. Furthermore, we have done some case studies on the one hand how to extend the MetaModelbrowser, e.g., for creation, update, and deletion of model elements as well as supporting model weaving, and on the other hand how to incorporate the MetaModelbrowser in current versioning systems.

1 Introduction

With the rise of model-driven development, model repositories are intended to facilitate research in model engineering and consequently in domain-specific modeling. Model repositories are central places where all kinds of modeling artifacts (e.g., meta-metamodels, 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 (typically necessary for domain-specific modeling languages) and for exchanging models, as well as a resource for teaching/learning materials.

There have been started some intiatives for building model repositories, e.g., zoomm.org, www.kermeta.org/mrep, or the Atlas MegaModel Management (AM3) [1]. The latter one is hosted within the popular Eclipse environment and is a subproject of the Generative Modeling Technologies (GMT) 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 [4]. The AM3 zoos are continuously growing and provide a respectable source of information in the meantime.

However, a more popular way of storing and organizing modeling artifacts is probably

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ecore:EPackage xmi:version="2.0"
   xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="SimpleUML"
   nsURI="http://SimpleUML" nsPrefix="SimpleUML">
 <eClassifiers xsi:type="ecore:EClass" name="UMLModelElement" abstract="true">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="kind" ordered="false" unique="false"
     lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name" ordered="false" unique="false"
     lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
 </eClassifiers>
 <eClassifiers xsi:type="ecore:EClass" name="Attribute" eSuperTypes="#//UMLModelElement">
  <eStructuralFeatures xsi:type="ecore:EReference" name="owner" ordered="false"
     lowerBound="1" eType="#//Class" eOpposite="#//Class/attribute"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="type" ordered="false" lowerBound="1"
     eType="#//Classifier" eOpposite="#//Classifier/typeOpposite"/>
 </eClassifiers>
[…]
```

Figure 1: XMI serialized model.

having a CVS like server software at hand. These repositories provide all means necessary to handle different versions of textual artifacts and let them compare syntactically. It seams therefore obvious to use existing tools also to store models as they can be serialized into the XMI format. Although versioning of models remains an open and challenging research field [11, 12], basic support for versioning models can be provided by CVS. CVS repositories can also be easily accessed via a standard Web browser providing an easy to use interface to users not familiar with IDE's such as Eclipse.

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. For example the Eclipse Modeling Framework's (EMF) [5] allows the user to view models within a tree-viewer in its *Sample Ecore Model Editor*. For a visually more appealing presentation of models having a concrete graphical syntax defined one can use the Graphical Modeling Framework (GMF) [6] to view these models. For Ecore based metamodels such a definition and editor comes along with the GMF plug-in.

Neither viewing models in the browser nor extra downloading them is always satisfactory. On one hand the textual representation of models is hardly readable and understandable for humans. In Figure 1, we depict a snippet of an Ecore based metamodel modeling the core of UML class diagrams. This representation of models is hardly readable because of type information from the Ecore meta-metamodel itself. This is further aggravated by the structure of the XML file representing only the containment structures of the Ecore meta-metamodel and not the structure of the metamodel. Also, the heavy use of XPath expressions complicates the otherwise easy to read metamodel. Note, that we refer sometimes to both model and metamodels when we talk about models. Only if we think it is necessary to distinguish explicitly between different levels of the OMG's 4-layer metamodeling stack we will do so.

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. This is the case for properties of a metamodel, which have been defined in its corresponding meta-metamodel, responsible for fine tuning the semantics of a metamodel.

Visualization of models in the context of Web browsers is not only desirable for the search in model repositories but also for any attempts building a Web application for model management purposes relying on models accessible through the WWW.

2 Web 2.0 to the Rescue

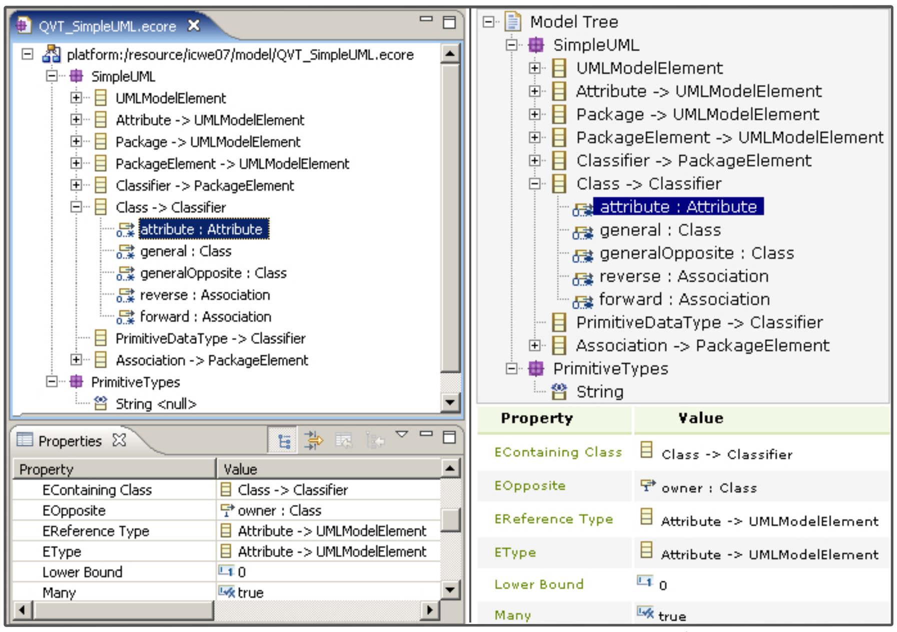
As a solution to the above presented problems, we propose our Web 2.0 MetaModelbrowser (MMB) as an alternative way of visualizing models within online model repositories. This MMB offers an Ajax-based tree-viewer for efficiently browsing Ecore-based metamodels and their models. In this respect, our goal 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. Based on the visualization functionality of our MMB we will demonstrate in a small case study, how the MMB can be placed into the context of model management in general.

Thus, our idea is to reproduce the user interface of EMF's Sample Ecore Model Editor (cf. Figure 2 (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 2 (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.

Benefits of our MMB are briefly summarized in the following:

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.


QVT_SimpleUML.ecore
platform:/resource/icwe07/model/QVT_SimpleUML.ecore
SimpleUML
UMLModelElement
Attribute -> UMLModelElement
Package -> UMLModelElement
PackageElement -> UMLModelElement
Classifier -> PackageElement
Class -> Classifier
attribute : Attribute
general : Class
generalOpposite : Class
reverse : Association
forward : Association
PrimitiveDataType -> Classifier
Association -> PackageElement
PrimitiveTypes
String <null>
Properties
Model Tree
SimpleUML
UMLModelElement
Attribute -> UMLModelElement
Package -> UMLModelElement
PackageElement -> UMLModelElement
Classifier -> PackageElement
Class -> Classifier
attribute : Attribute
general : Class
generalOpposite : Class
reverse : Association
forward : Association
PrimitiveDataType -> Classifier
Association -> PackageElement
PrimitiveTypes
String
(a)
(b)

Property	Value
EContaining Class	Class -> Classifier
EOpposite	owner : Class
EReference Type	Attribute -> UMLModelElement
EType	Attribute -> UMLModelElement
Lower Bound	0
Many	true

Figure 2: SimpleUML metamodel in (a) Eclipse and (b) MMB.

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 2 (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 2).

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 3.3).

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.

Exchange. We allow the user to have an alternative representation of models he/she can view per URI. This can ease the exchange of those models among partners worldwide. With developers or customers located around the world this leads to more efficient distribution of knowledge and information.

Model Management. The browsing and viewing of models is only one particular aspect of exploring online repositories, of course. Other features are needed to allow for substantial benefit from Ajax-based Web applications. As to this we present in Section 4.1 a case study of online model weaving, that demonstrates how the MMB can be extended to get one step closer to a desktop like model engineering tool.

Support for Versioning Systems. The integration of our MMB in existing versioning system browsers, e.g., for browsing SVN repositories, allows to use our service as a front-end for current state-of-the-art versioning systems in which metamodel and model versions are stored. Of course, various additional features for browsing different versions of metamodels and models could be implemented such as a diff-viewer for showing the differences between two metamodels graphically.

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 Prerequisites for MMB

Before we will go into implementation details we will give a brief overview of the framework, that constitutes our MMB, see Figure 3. As already mentioned our MMB operates on Ecore-based metamodels. Using the OMG's terminology we rely on the M3 layer on Ecore as a meta-metamodel. Based on this meta-metamodel we can handle all to Ecore conforming metamodels. Viewing of models is of course then restricted to models conforming to some well defined metamodels. These modeling artifacts have been created in the user workbench by some tool and need to be serialized in XMI format. To be able to use those models in combination with out MMB they have to be stored in some repository that is accessible through HTTP. The MMB can retrieve the models by a simple request and present them to the user. The view service for metamodels is a generic feature as the meta-metamodel is not a parameter when invoking the view request. Making the meta-

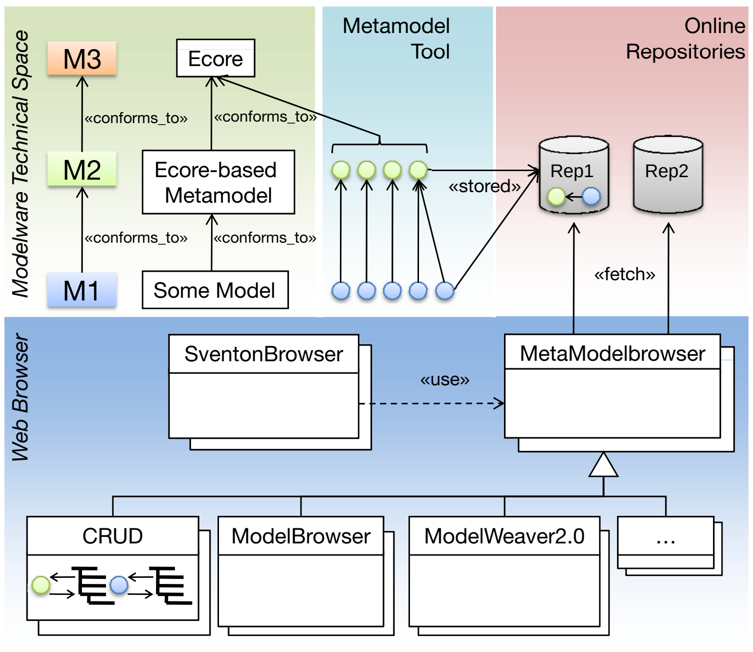


Figure 3: Overview of the MMB framework.

metamodel a variable would require to build a completely new framework for modeling, as EMF solely relies on Ecore and does not for example support MOF specified by the OMG. To view models of some modeling language is not supported in a generic way. Right now an additional plug-in for our MMB has to be generated within Eclipse by our Model Browser component manually, see Sections 3.3 and 5. The functionality of the MMB can be further extended by particular components like one that supports all four CRUD (Create, Retrieve, Update, Delete) operations. Components that support model weaving will be discussed later.

3.2 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 [7]. As is shown in Figure 4 (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

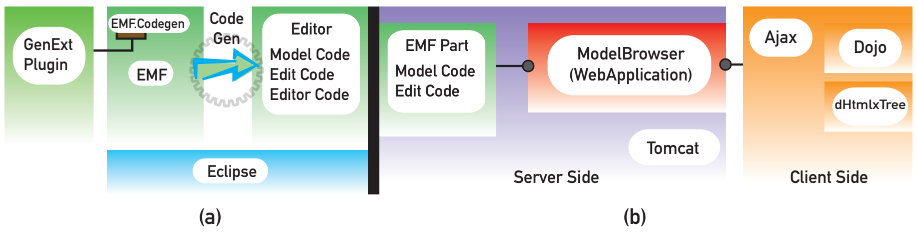


Figure 4: EMF components overview (a) and MMB architecture (b).

Eclipse workbench.

The MMB architecture, depicted in Figure 4 (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 [8] and Dojo [9] as shown in Figure 4 (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.

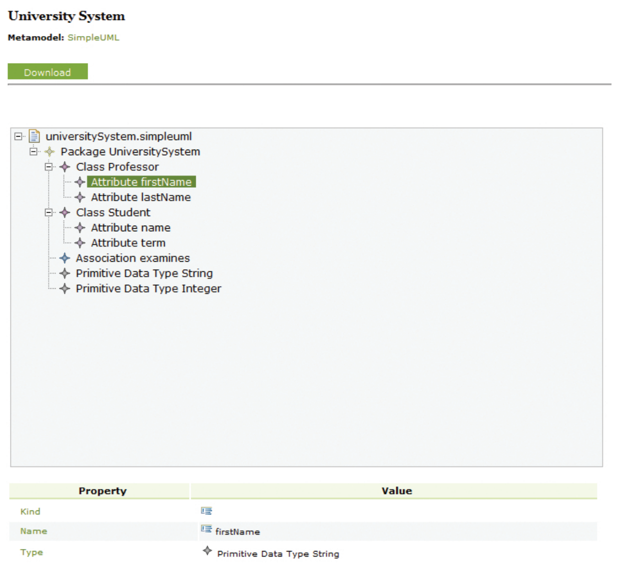
3.3 Browsing Models

In the above section we described how to browse metamodels by reusing existing EMF components and integrating them within the Web application. More challenging, 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 4 (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.

A screenshot of browsing a SimpleUML model is shown in Figure 5.

4 Extending the MetaModelbrowser Architecture

In this section, we present two case studies how to extend our browsing capabilities for metamodels and models.


University System

Metamodel: SimpleUML

Download

```
universitySystem.simpleuml
  Package UniversitySystem
    Class Professor
      Attribute firstName
      Attribute lastName
    Class Student
      Attribute name
      Attribute term
    Association examines
    Primitive Data Type String
    Primitive Data Type Integer
```

Property	Value
Kind	
Name	firstName
Type	Primitive Data Type String

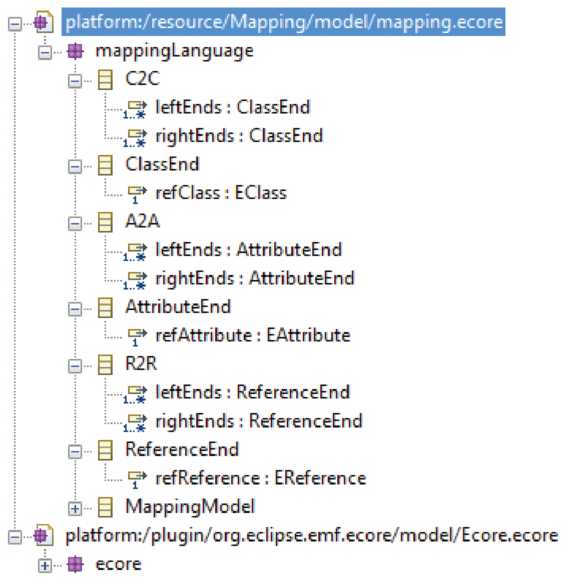
Figure 5: Browsing models conforming to a known metamodel.

The first case study deals with model management which is an important task in the software development process. Model operations therefore play a major role and are vital if the model engineering field shall be lifted to the Web. In the following we will present our early prototype ModelWeaver 2.0, which realizes tasks such as model creation, model update, model element deletion and model weaving.

The second case study is about how to incorporate our browsing capabilities in already existing versioning system browsers. The basic view service of our MMB primarily targets models residing in a simple Web directory. In order to provide also access to CVS model repositories which store metamodel and model versions, we have developed a simple extension to Sventon [2], a Java application capable of browsing Subversion repositories.

4.1 ModelWeaver 2.0

Based on the ATLAS Model Weaver (AMW) [14] and the MMB we have implemented a prototype, that enables the user to weave any two metamodels within the Administration Tool of the MMB. Note, that the MMB provides a simple repository by itself, which is however not open to the public yet, as we have not implemented any access control model. Our ModelWeaver 2.0 [22] builds again on the EMF to provide a mapping metamodel and let the user create, update, and browse a mapping model, that describes the dependencies between the elements of two modeling languages, i.e., metamodels. We started out with a simple mapping metamodel depicted in Figure 6. This mapping model is quite similar to



```
platform:/resource/Mapping/model/mapping.ecore
  mappingLanguage
    C2C
      leftEnds : ClassEnd
      rightEnds : ClassEnd
    ClassEnd
      refClass : EClass
    A2A
      leftEnds : AttributeEnd
      rightEnds : AttributeEnd
    AttributeEnd
      refAttribute : EAttribute
    R2R
      leftEnds : ReferenceEnd
      rightEnds : ReferenceEnd
    ReferenceEnd
      refReference : EReference
    MappingModel
platform:/plugin/org.eclipse.emf.ecore/model/Ecore.ecore
  ecore
```

Figure 6: Mapping Metamodel for the ModelWeaver 2.0

the core weaving metamodel of the AMW. However it is more specialized in the sense that we do not define a generic weaving link, able to connect any two modeling elements. The mapping ends types are determined by the types Ecore specifies for creating metamodels. For example the *ClassEnd*s contained in a Class2Class (*C2C*) mapping are always of type *EClass*, which is defined in Ecore. Our mapping metamodel can not be strictly classified as M2 model according to the OMG's 4-layer architecture. As can be seen at the bottom of Figure 6 Ecore itself is referenced as resource into our mapping metamodel. This is because we need to access specific types of Ecore in our metamodel, which makes it in fact also an extension to Ecore, our meta-metamodel. From these perspectives it is not just a simple metamodel, although we will pretend in the following that it is just a pure metamodel.

Figure 7 shows the main view of the ModelWeaver 2.0. The view service of the MMB has been completely reused in order to visualize the two metamodels positioned on the left and the right as well as the mapping model which is located in the middle. In Figure 7 we illustrate a simple weaving application with our SimpleUML and a basic Entity Relationship language. Every concrete mapping model has a root element called *MappingModel* according to the mapping metamodel. This model element is instantiated automatically upon the start of the weaving. In a next step the user selects this element and chooses one of the three basic mapping operators available (*C2C, A2A or R2R*) from the drop down menu. Afterwards he/she can add the chosen element to the mapping model and create left and right ends for this mapping operator. Because of the drag&drop support of the dHtmlxTree Ajax framework it is then possible to drag one element from the left or right hand

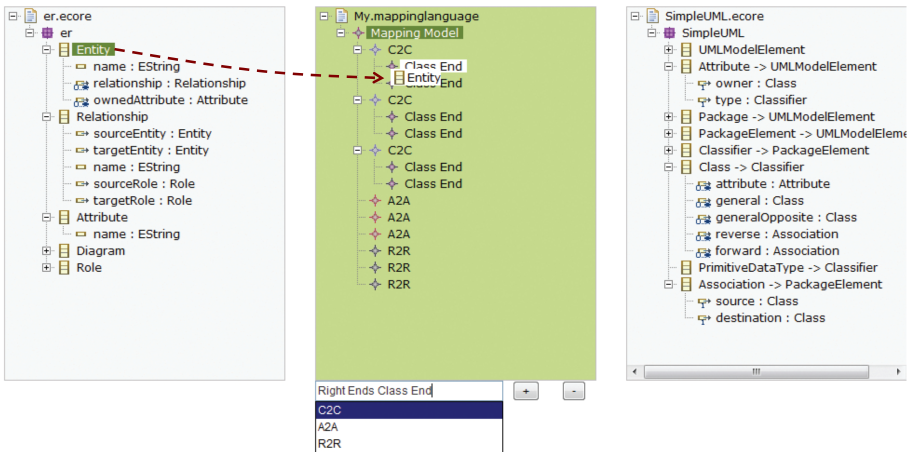


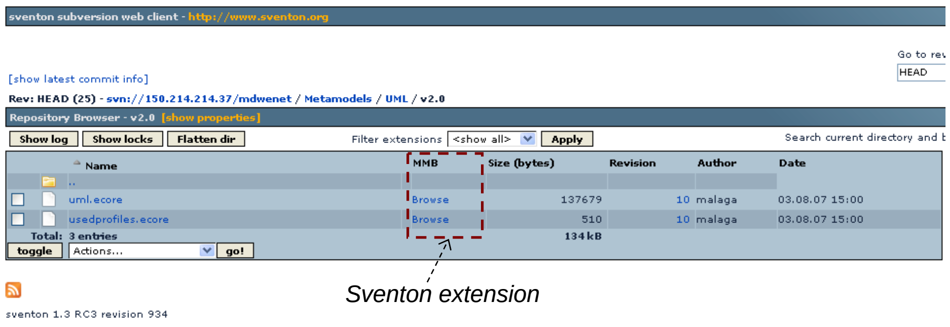
Figure 7: Weaving view of the ModelWeaver 2.0

side tree and drop it to a suitable mapping operators end. Also the deletion and modification of mapping model elements is supported. When the weaving task is finished the user has the possibility to download the created mapping model for later usage or the usage in another development environment to enable for example the creation of a transformation model. The derivation of transformation models based on the metamodel weavings is not implemented yet and remains an open issue for future work. If the user decides to get back to work later on it is also possible to load an existing mapping model into the next session provided that the same metamodels have been selected for the weaving task.

4.2 SventonBrowser

The most common versioning systems for documents of any kind are most likely the Concurrent Versions System (CVS) and Subversion (SVN). These systems are also used to store models to have at least the possibility to trace changes between different versions of models and compare them if necessary on a textual level such as the XMI serialization of the metamodels and models. To handle data stored within such versioning systems with our MMB framework, e.g. view these models with the visualization service, a Web interface is needed. An application providing such an interface is the Java-based Web application called Sventon, which allows to browse SVN repositories. Provided that all information needed to access an SVN repository is given, proper URLs can be passed the MMB as input to view the stored metamodels. Constructing URLs and passing them correctly to our MMB application in order to view one's own metamodels is however not very user-friendly. Due to this we have implemented a small extension to the Sventon application, which integrates our visualization service by constructing the right URLs. Our "Sventon Browser" is available under the URL:

www.metamodelbrowser.org/sventon


sventon subversion web client - http://www.sventon.org

Go to rev
HEAD

[show latest commit info]

Rev: HEAD (25) - svn://150.214.214.37/mdwenet / Metamodels / UML / v2.0

Repository Browser - v2.0 [show properties]

Show log   Show locks   Flatten dir   Filter extensions <show all>   Apply   Search current directory and b

		Name	MMB	Size (bytes)	Revision	Author	Date
		..					
☐		uml.ecore	Browse	137679	10	malaga	03.08.07 15:00
☐		usedprofiles.ecore	Browse	510	10	malaga	03.08.07 15:00
	Total:	3 entries		134 kB			

toggle   Actions...   go!

sventon 1.3 RC3 revision 934

Figure 8: SventonBrowser using MMB

Unfortunatly only one repository can be configured over the Sventon Web interface at the moment. Manual entries can however be achieved through properties files in the Tomcat directory. Figure 8 shows a screenshot of our extended Sventon, that illustrates the functionality for the MDWEnet model repository. The URL's for any Web accessible repository, especially for Sventon, look like this:

```
www.metamodelbrowser.org/BrowseTreeServlet?
url=http://metamodelbrowser.org:80/
sventon/showfile.svn?path=<URI_OF_MODEL>&<some_parameters_needed>
```

With the conduction of this small case study we demonstrate how existing repositories based on CVS or SVN can be accessed and browsed using the MMB. This allows now a greater audience to benefit from our MMB framework, which might be in the near future not only limited to the model view operation.

5 Critical Discussion

Our MMB supports the visualization of models through the ModelBrowser plug-in. This plug-in can only be used within a running Eclipse workbench to generate the necessary model code for the Web application as described above. This solution is not very comfortable and undermines our efforts to build a platform independent application aiming at model management in general. Thus, it would be nice to have a fully automatic generation of Modelbrowser code. This means, for the user it should be possible to upload an Ecore-based metamodel and the Modelbrowser code, which allows visualizing models conforming to the metamodels, is automatically generated by the Web application. The user no longer needs to activate the code generation within EMF and upload the files to the Web server. We discovered, that this code generation is not so simple to integrate with a standalone Java application. The reason for this is that EMF uses JET (Java Emitter Templates) [3]. JET needs to be invoked within an Eclipse workbench. According to our research there seem to be workarounds to handle code generation in a standalone manner, but this would also require at least an Eclipse ”Driver” available on the server side.

6 Related Work

With respect to our MetaModelbrowser framework we identified three major fields of related work.

Visualization of models. There has been a lot of work done in the area of model visualization. Especially related to us is the two dimensional graphical representation of software models. Automatic layout approaches for UML class diagrams considering certain kinds of aesthetic criteria are for example presented in [15], [16]. Based on the work on planar graphs, i.e., graphs that have no crossing edges, our MetaModelbrowser could be extended to support the visualization of large class diagrams.

Model Repositories. The idea of storing data as well as software building artifacts and their management dates back to the early 90s. With the Portable Common Tool Environment (PCTE) [17] a framework was developed and should integrate popular software engineering tools and support the management of large data. Today, the basis for the success of model repositories is certainly an accepted common interchange format like XMI. However, the goal of interoperability is not yet achieved. The ModelCVS project [19] therefore tries to overcome interoperability problems among different tool vendors and to provide a framework for model management.

Also, the idea of model repositories goes far beyond the software engineering community. In bio engineering, the modeling of complex organic structures becomes more and more important. Also the exchange of these models among various stakeholders gets crucial. The cellML [18], an XML-based model interchange format and repository with over 300 models, is a perfect example for the importance of online repositories and the appropriate visualization of the stored models.

Web 2.0. The social aspects of the Web 2.0 movement, such as participation in an informal manner, higher usability due to AJAX technology, and the lightweightness, have been its key for success. Braun et al. [20] also base their work on Web 2.0 to allow for a collaborative informal ontology engineering process. For their Web based implementation SOBOLEO, a lightweight ontology editor and social bookmarking system, they also incorporate AJAX technology. If we consider metamodels somehow equivalent to ontologies [21] it is possible to extend our MetaModelbrowser to act as a lightweight collaborative metamodel editor.

7 Conclusions and Future Work

In this work we have presented an Ajax-based approach to visualize metamodels and models on the Web. The approach has been implemented as a public visualization service based on existing Eclipse technologies. This visualization service is now used in the ModelCVS project [10], in the MDWEnet project [13], and for visualizing about 250 metamodels of the AtlantEcore Zoo[1].

[1]http://www.eclipse.org/gmt/am3/zoos/atlantEcoreZoo/

Also we have demonstrated by a well known model engineering technique how our MMB can be extended into a framework supporting various kinds of model management operators and how the visualization service can be integrated into existing SVN repository viewers.

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 inital MMB allows read-only access to models but could be extended for editing functionality similar to the model weaving case study. 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.

8 Acknowledgments

We thank Jeremy Solarz for his support during implementation of the ModelWeaver 2.0.

References

[1] ATLAS MegaModel Management, official site: http://www.eclipse.org/gmt/am3

[2] Sventon. Java Repository Browser, official site: http://www.sventon.org/

[3] JET. Java Emitter Templates, Model2Text Transformation, official site: http://www.eclipse.org/modeling/m2t/?project=jet

[4] Allilaire, F., Bézivin, J., Bruneliére, H., and Jouault, F.: Global Model Management in Eclipse GMT/AM3. Eclipse Technology eXchange workshop in conjunction with ECOOP'06, 2006

[5] Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.: Eclipse Modeling Framework. Addison Wesley, 2003

[6] Graphical Modeling Framework (GMF), official site: http://www.eclipse.org/gmf

[7] OMG: Meta Object Facility (MOF) 2.0 Core Specification, 2003

[8] dhtmlxTree - AJAX powered DHTML JavaScript Tree component with rich API, official site: http://scbr.com/docs/products/dhtmlxTree/

[9] Dojo - The Javascript Toolkit, official site: http://dojotoolkit.org/

[10] Kappel, G., Kapsammer, E. , Kargl, H. , Kramler, G., Reiter, T., Retschitzegger, W., Schwinger, W., Wimmer, M.: On Models and Ontologies - A Layered Approach for Model-based Tool Integration, Modellierung 2006, Innsbruck, March 2006, GI LNI 82, pp. 11-27, 2006.

[11] Reiter, T., Altmanninger, K., Bergmayr, A., Schwinger, W., Kotsis, G.: Models in Conflict - Detection of Semantic Conflicts in Model-based Development. In 3rd International Workshop on Model-Driven Enterprise Information Systems (MDEIS-2007), Funchal, Madeira - Portugal, June 2007.

[12] France, R., Rumpe, B.: Model-driven Development of Complex Software: A Research Roadmap. FOSE '07: 2007 Future of Software Engineering, IEEE Computer Society, pp. 37-54, 2007.

[13] Vallecillo et al.: MDWEnet: A Practical Approach to Achieving Interoperability of Model-Driven Web Engineering Methods. In Workshop Proc. of 7th Int. Conference on Web Engineering (ICWE'07), Como, Italy, July 2007, Politecnico di Milano, pp. 246-254, 2007.

[14] Del Fabro, M.D., Bézivin, J., Jouault, F., Gueltas, G.: AMW: A Generic Model Weaver, Proceedings of the 1ère Journée sur l'Ingénierie Dirigée par les Modèles (IDM05), 2005.

[15] Gutwenger, C., Jünger, M., Klein, K., Kupke, J., Leipert, S., Mutzel, P.: A new approach for visualizing UML class diagrams, SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization, San Diego, California, pp. 179-188, 2003.

[16] Eichelberger, H.: Aesthetics and Automatic Layout of UML Class Diagrams, PhD Thesis, Julius-Maximilians-Universität Würzburg, 2005.

[17] Long, F., Morris, E.: An Overview of PCTE: A Basis for a Portable Common Tool Environment, Technical Report, Carnegie Mellon University, 1993.

[18] Cuellar, A., Lloyd, C., Nielsen, P., Bullivant, D., Nickerson, D., Hunter, P.: An Overview of CellML 1.1, a Biological Model Description Language, SIMULATION: Transactions of The Society for Modeling and Simulation International, 79(12):740-747, 2003.

[19] Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., Schwinger, W., Wimmer, M.: On Models and Ontologies - A Layered Approach for Model-based Tool Integration , Proceedings of Modellierung 2006, GI-Edition, Lecture Notes in Informatics, Eds.: H.C. Mayr, R. Breu, 22-24 March, Innsbruck, Austria, 2006.

[20] Braun, S., Schmidt, A., Walter, A., Nagypal, G., Zacharias, V.: Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering, Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at the 16th International World Wide Web Conference (WWW2007) Banff, Canada, May 8, 2007.

[21] Kappel, G., Kargl, H., Kramler, G., Reiter, T., Schwinger, W., Wimmer, M.: Lifting Metamodels to Ontologies: A Step to the Semantic Integration of Modeling Languages, ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML 2006), Genova, Italy, October 2006.

[22] Solarz, J.: Model Weaver 2.0: Eine AJAX-basierte Webanwendung für Model Weaving, Master Thesis, Vienna University of Technology, 2008 (in German).