Demonstration of a Tool for Consistent Three-way Merging of EMF Models

Felix Schwägerl, Sabrina Uhrig and Bernhard Westfechtel

University of Bayreuth, Universitätsstr. 30, 95440 Bayreuth, Germany
{felix.schwaegerl,sabrina.uhrig,bernhard.westfechtel}@uni-bayreuth.de

Abstract. Version control systems have become indispensable in contemporary software development. Optimistic strategies allow for independent modifications of the same software artifact, e.g. model. As soon as these modifications happen concurrently, three-way merging comes into play. Merging tools specific to the requirements of model-driven development are urgently needed. We demonstrate a three-way merge tool for EMF models which supports the user committing his decisions in order to resolve merge conflicts. The user interface consists of an editor that shows the superimposition of the three input versions and a wizard that guides the user through the resolution of pending merge conflicts. In our example scenario we merge concurrent modifications on a medium-sized UML model.

1 Background

Inadequate version control has been identified as a major obstacle to the application of model-driven software engineering. In particular, sophisticated support for *merging model versions* is urgently needed. Line-oriented or structure-oriented (e.g. XML-based) merging tools do not address crucial requirements concerning the consistency of merge results and the detection and resolution of merge conflicts to a sufficient extent.

In [4], we have developed a formal approach to state-based three-way merging of model versions in the *Eclipse Modeling Framework* (EMF [3]). It allows to detect and resolve *context-free conflicts* occurring on the same structural feature of some EMF object, as well as *context-sensitive conflicts* considering changes of different structural features of potentially different objects. Our approach advances the state of the art by guaranteeing a merge result that is consistent with the structural constraints imposed by EMF and by allowing for merging objects from different classes. We only require that the three models are instances of the same Ecore model. The implementation follows an incremental design allowing to pause or revert merge decisions at any time.

After a brief tool overview in Section 2, we present our demonstration plan in Section 3. We conclude with installation instructions and a web link to the screencasts.

2 Tool Overview

Our merge tool itself is based on EMF; the *BTMerge* metamodel defines the structure of *merge models* which represent the superimposition of the three EMF model versions involved in a merge. As shown in Figure 1, merging with our tool is a three-phase process:

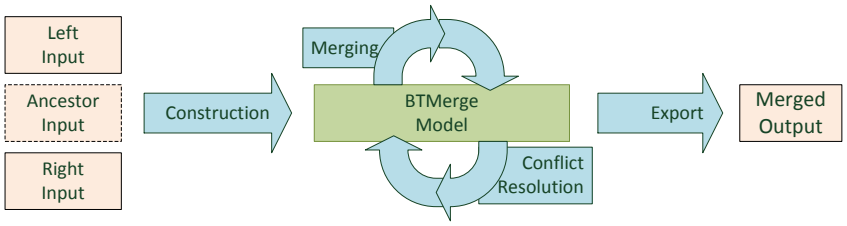

Fig. 1. Conceptual overview on our merge tool.

First, the merge model is created (*construction*). Corresponding EMF objects are identified either by unique identifiers (*UUID*s) or by means of a matching algorithm, e.g. the *EMF Compare match engine* [1]. The second phase, *merging*, follows an incremental design: The preliminary merge model is modified alternately by the merge algorithm and the user; the merge algorithm applies *merge rules* which can either be applied automatically or require user interaction in case of *conflicts*. Only after all conflicts are resolved, the merge model is ready to be *exported* back into an EMF instance.

The user interface, the *resolution tool* (cf. screenshot Fig. 2), is based on a generated EMF tree editor and allows the user to communicate resulution decisions for specific conflicts during the second phase. The *main* editing view (right top) shows the containment tree of the model versions in multiple columns. Below, a modified *properties* view reflects values of structural features of the superimposed object selected in the main view. Conflicting objects and values are marked by icon overlays. The *conflicts* view located at the left bottom outlines pending merge decisions. The user can resolve a conflict by double-clicking on it, which opens a wizard that will describe the conflict and propose several resolution methods. After resolution by the user, the next merge increment is performed automatically until the *merging* phase terminates.

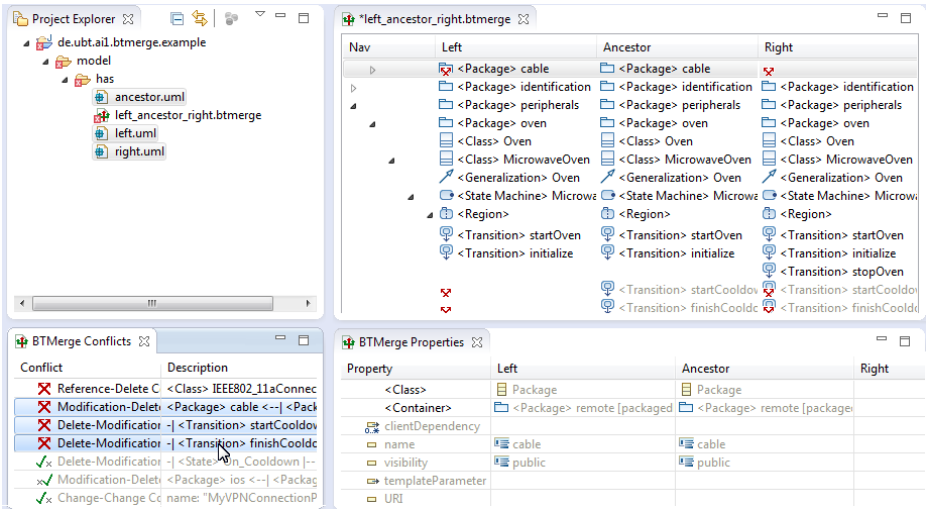

Fig. 2. A screenshot of the resolution tool with the “Home Automation System” example.

3 Demonstration Plan

We apply our merge tool to a medium-sized scenario ("Home Automation System") where three revisions of the same UML [2] model, a common *ancestor* and two alternative versions (*left* and *right*), which result from concurrent modifications, are merged. The plan is accompanied by our screencasts (web link, see Section 4).

1. **Applying Concurrent Modifications**. By means of the UML tree editor, the following conflicting modifications are applied to two copies of the same model:
   (a) Renaming of a class `VendorVPNConnectionProvider` to `MyVPNConnectionProvider` (left) vs. `YourVPNConnectionProvider` (right).
   (b) Insertion of a generalization to class `IEEE802_11aConnector` (left) vs. deletion of the corresponding class (right).
   (c) Visibility change of package `ios` (left) vs. deletion of that package (right).
2. **Merging**. An initial merge model is created from the three versions using the provided context menu entry. In subsequent increments, the following conflicts are reported to the user and resolved by means of the resolution wizard:
   (a) *Change-Change Conflict* on the feature `name` of `VendorVPNConnectionProvider`. The user selects value `MyVPNConnectionProvider` (left).
   (c) *Reference-Delete Conflict* on class `IEEE802_11aConnector`. The user selects "reference" (left) and discards the deletion (right).
   (b) *Modification-Delete Conflict* on package `ios`. The user applies the deletion (right), discarding the modification (visibility change, left). The resolution of this conflict leads to the automatic resolution of related conflicts caused by it.
3. **Result**. The merge output is a valid EMF instance and contains all modifications that have been selected by the user. Furthermore, non-conflicting changes have been applied (e.g. inside the `MicrowaveOvenControl` state chart).

4 Installation Instructions and Screencasts

Our software is available on an update site[1] and can be installed as plug-ins into a clean *Eclipse Modeling Tools* distribution (Juno or higher). Screencasts demonstrating both the installation and the usage of our tool are provided on our web pages[2].

References

1. Brun, C., Pierantonio, A.: Model differences in the Eclipse Modelling Framework. UPGRADE IX(2), 29–34 (Apr 2008)
2. OMG: OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.3. OMG, Needham, MA, formal/2010-05-05 edn. (May 2010)
3. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF Eclipse Modeling Framework. The Eclipse Series, Addison-Wesley, Upper Saddle River, NJ, 2nd edn. (2009)
4. Westfechtel, B.: Merging of EMF models: Formal foundations. Software and Systems Modeling p. 32 p. (Oct 2012), http://dx.doi.org/10.1007/s10270-012-0279-3, Online First

[1] http://btn1x4.inf.uni-bayreuth.de/btmerge/update/
[2] http://btn1x4.inf.uni-bayreuth.de/btmerge/screencasts/