Conflict Resolution in Model Versioning*

Petra Brosch, Konrad Wieland, and Gerti Kappel

Business Informatics Group, Vienna University of Technology, Austria
{lastname}@big.tuwien.ac.at

Abstract. Optimistic version control systems enable globally distributed teams of developers to work together asynchronously. Every developer works on a local copy and consequently, no developer is ever detracted from working by waiting for a resource. The price for this flexibility is payed at the moment when conflicting modifications must be integrated into one consolidated version.
In this paper, we discuss conflicts and their need for resolution in the context of model versioning and provide the basic concepts necessary to build a model versioning system which guides modelers through the critical consolidation phase by recommending suitable patterns.

1 Introduction

When multiple developers concurrently work on the same artifact, conflicting modifications are very likely to be performed. Merging the different versions manually poses a very time-intensive, repetitive challenge in order to obtain a consistent artifact which meaningfully integrates the work of all involved developers. Approaches for automating the merge process include the calculation of all possible combinations of parallel performed operations leading to a valid version and merge policies to privilege specific operations or operations of specific users (cf. [4] for a survey). These approaches only ask for user input in undecidable cases and reduce the manual resolution effort to a minimum. However, they neither consider resolution strategies that go beyond a recombination of conflicting changes, nor the need for temporarily tolerating conflicts [2]. In some situations conflicts may be resolved more reasonable by providing a completely new version or by including conflicting changes in the model for resolving them collaboratively in further iterations.

To support the merging process of software models, we present a recommender system which suggests automatically executable conflict resolution patterns derived from the resolution of similar conflict situations already occurred. The Resolution Recommender is integrated into the adaptable model versioning system AMOR [1], which reports not only conflicts resulting from overlapping atomic changes, but also conflicts related to composite operations like refactorings which imply a multitude of possible resolution approaches.

* This work has been partly funded by the Austrian Federal Ministry of Transport, Innovation, and Technology and the Austrian Research Promotion Agency under grant FIT-IT-819584 and by the fFORTE WIT Program of the Vienna University of Technology and the Austrian Federal Ministry of Science and Research.

2 Conflict Resolution in Model Versioning

A prerequisite for the recommendation of conflict resolution patterns is the precise description of the conflict situation. A conflict is composed by the involved model elements, the performed changes as well as the violated constraints. These constraints are either preconditions of a change or conformance rules defined in the metamodel of a modeling language. The Resolution Recommender looks up the Resolution Pattern Storage for suitable resolution patterns, matching the conflict description of the current situation. The resolution patterns in this repository are either defined manually or are automatically mined as described in [3]. If no resolution patterns were found, a manual resolution has to be performed, which acts in turn as input for the Resolution Recommender to infer new resolution patterns. Whenever resolution recommendations are at hand, they are ranked by relevance and presented to the user. The proposed resolution patterns may be previewed, rolled back, and manually refined. In both, manual and semi-automatic conflict resolution, a special *Conflict Diagram View* is created. This view provides an integrated prospect of the model's evolution by applying non-conflicting changes, marking pending conflicting changes, and annotating all involved elements with special stereotypes defined in the Conflict Profile, available on our project website[1]. Despite including conflicts, the Conflict Diagram View ensures a valid model, which may be displayed in any UML editor or may be committed to the model versioning system to postpone conflict resolution.

Next Steps. We are currently working on similarity measures of metamodel elements to identify conflict resolution candidates not exactly fitting the situation in place. In order to expand the applicability of inferred resolution patterns to similar situations, we elaborate higher-order transformations [5] to adapt resolution patterns. An orthogonal approach we are working on, includes establishing a collaborative setting for the resolution of deferred conflicts.

References

1. P. Brosch, G. Kappel, P. Langer, M. Seidl, K. Wieland, M. Wimmer, and H. Kargl. Adaptable Model Versioning in Action. In *Modellierung 2010*, pages 221–236, 2010.
2. P. Brosch, P. Langer, M. Seidl, K. Wieland, and M. Wimmer. Concurrent Modeling in Early Phases of the Software Development Life Cycle. In *16th Conf. on Collaboration and Technology*, 2010, to appear.
3. P. Brosch, M. Seidl, and M. Wimmer. Mining of Model Repositories for Decision Support in Model Versioning. In *Proc. of the Europ. MDTPI Workshop*, pages 25–33. CTIT Workshop Proceedings, 2009.
4. T. Mens. A State-of-the-Art Survey on Software Merging. *IEEE Transactions on Software Engineering*, 28(5):449–462, 2002.
5. M. Tisi, F. Jouault, P. Fraternali, S. Ceri, and J. Bézivin. On the Use of Higher-Order Model Transformations. In *Proc. of the 5th Europ. Conf. on Model Driven Architecture-Foundations and Applications*, pages 18—33. Springer, 2009.

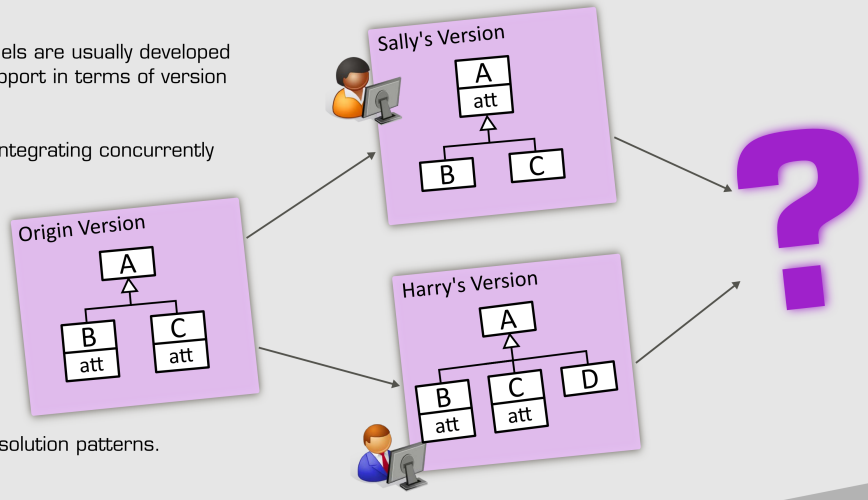
[1] `http://www.modelversioning.org/conflict-profile`

Petra Brosch, Konrad Wieland, and Gerti Kappel

Conflict Resolution in Model Versioning

Conflict

- Like traditional code, software models are usually developed in teams requiring collaboration support in terms of version control systems (VCS).
- One use case of such a system is integrating concurrently evolved versions of one model into one consistent version.
- When the modifications are contradicting, then the VCS reports the conflict, but the cumbersome resolution process is left to the user.
- We present a recommender system which suggests automatically executable conflict resolution patterns.



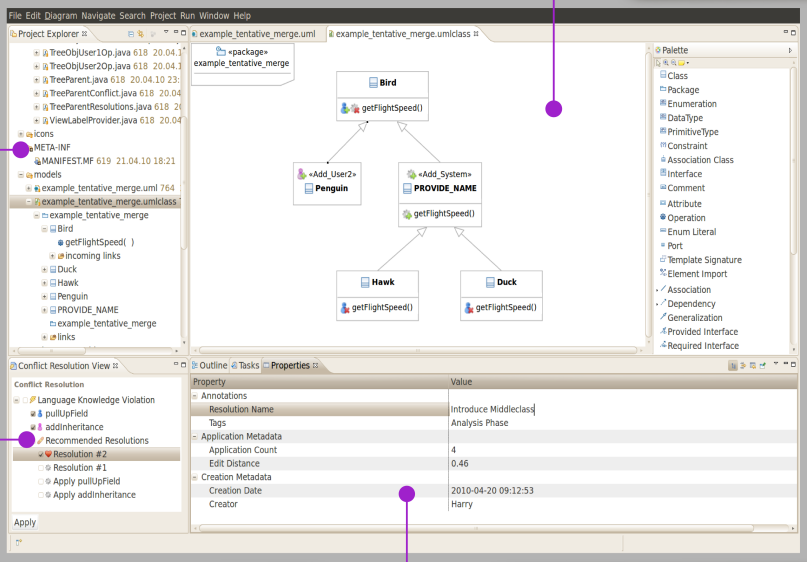
Resolution

Project Explorer

The Eclipse Team Support plugin yields the basis for the interplay with the versioning server.

Model Editor

The model editor offers the possibility to remodel artifacts to resolve conflicts. Model elements are marked with a dedicated symbol indicating changes.



Recommendations

The recommender system supports the conflict resolution by providing a list of automatically applicable resolutions patterns.

Resolution Properties

Resolution specific metadata is displayed in a dedicated property view.

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology
JKU JOHANNES KEPLER UNIVERSITY LINZ
SPARX SYSTEMS

AMOR - Adaptable Model Versioning
http://www.modelversioning.org

Partly funded by the Austrian Federal Ministry of Transport, Innovation and Technology and FFG under grant FIT-IT-819584, and by the fFORTE WIT Program of the Vienna University of Technology and the Austrian Federal Ministry of Science and Research.