AMOR – Towards Adaptable Model Versioning

Kerstin Altmanninger[1], Gerti Kappel[2], Angelika Kusel[1], Werner Retschitzegger[1], Wieland Schwinger[1], Martina Seidl[2], Manuel Wimmer[2]

[1]Johannes Kepler University Linz, Austria
{kerstin.altmanninger | angelika.kusel | werner.retschitzegger | wieland.schwinger}@jku.at
[2] Vienna University of Technology, Austria
{kappel | seidl | wimmer}@big.tuwien.ac.at

Abstract. The development of complex software systems requires appropriate abstraction mechanisms in terms of model-driven engineering techniques (MDE) and proper support for allowing developers to work in parallel in terms of version control systems (VCSs). For realizing the vision of MDE, a bundle of standards has been made available recently, whereas the versioning of models has not gained the necessary attention yet, although being of paramount importance for the success of MDE in practice.
In this paper, we propose a first vision of AMOR (Adaptable Model Versioning) to leverage version control in the area of MDE. The innovations of AMOR are threefold. Firstly, AMOR supports precise conflict detection, i.e., previously undetected as well as wrongly indicated conflicts shall be avoided. Secondly, AMOR focuses on intelligent conflict resolution by providing techniques for the representation of conflicting modifications as well as suggesting proper resolution strategies. Thirdly, AMOR targets an adaptable versioning framework, empowering modelers to flexibly balance between reasonable adaptation effort and proper versioning support while ensuring generic applicability to various domain-specific modeling languages and associated tools.

Keywords: Model Versioning, Model Evolution, Model Comparison, VCS

1 Introduction

Versioning in MDE. The development of complex software systems requires appropriate abstraction mechanisms in terms of model-driven engineering techniques (MDE) and proper support for allowing developers to work in parallel in terms of version control systems (VCSs). Regarding the area of MDE, models are placed as first class artifacts throughout the software life-cycle, allowing step-wise code generation on the basis of systematic transformations [1]. For realizing the vision of MDE, a bundle of standards has been made available recently. Considering the field of VCSs, a multitude of solutions for code versioning has been proposed in the last four decades like CVS [2] or Subversion [3], supporting detection and resolution of conflicting modifications. Versioning of models, however, has not gained the necessary attention yet, although being of paramount importance for the success of MDE in practice.

Problems with line-oriented Versioning. Simply taking code versioning techniques that are mostly line-oriented and applying them to XMI-serializations of models is not reasonable for *optimistic versioning* approaches which are indispensible in practice. In contrast to *pessimistic approaches* following the lock-modify-unlock paradigm, in optimistic versioning multiple modelers may check-out and modify the same artifact at the same time. This means that two versions of a model have to be *compared*, possible conflicts have to be *detected* and *resolved*, and finally a *merge* needs to be performed to obtain a consistent model which is again checked-in into a model repository. In line-oriented approaches, the graph-oriented nature of models and even more problematic, their rich semantics are totally neglected.

Focus of this Paper. Not properly considering the nature of models implicates three problems identified in Section 2, which hamper effective model versioning. Since current approaches do not appropriately tackle all of the three problems, we propose the system AMOR, an adaptable versioning framework. We present the conceptual architecture behind AMOR, and finally, we outline ongoing work concerning implementation, evaluation, and customization.

2 Problems in Model Versioning

The negative impact of inappropriate versioning support for models in MDE is problematic in three different ways:

(1) Erroneous Conflict Detection. Conflict detection, representing one of the core tasks during versioning, is error prone meaning that often either real conflicts are not detected or wrong conflicts are found. As a consequence of erroneous conflict detection, inconsistent and incomplete specifications occur, jeopardizing the correct modeling of the system under development.

(2) Unsupportive Conflict Resolution. Conflict resolution, another core task of versioning, is currently insufficiently supported as it is mostly within the sole responsibility of the user to repetitively interpret the rationale behind modifications, to identify possible conflict resolution strategies, and finally, to decide for applying a selected strategy. The neglect to deal with such repetitive conflict resolution situations results in decreased productivity, increased error rates, and potentially neglected cooperate design principles.

(3) Inflexible VCS. Existing VCSs are either *generic*, i.e., applicable to any modeling language, being characterized by poor versioning support, or they exhibit enhanced versioning support, tightly bound to a specific modeling language and often additionally to a specific modeling environment. This inflexibility is even worse because of the rapidly growing number of domain-specific modeling languages (DSMLs), entailing either the straightforward application of an existing generic system or requiring herculean efforts in developing a dedicated VCS from scratch.

Summarizing, these three obstacles aggravate the versioning of models and without appropriate solutions, model VCSs will not gain widespread acceptance by modelers. An important tool for a successful development process will be missing.

3 Related Work

Most recently, numerous model versioning systems have been proposed. In the following, we discuss how current systems handle the aforementioned problems.

Conflict Detection. In contrast to standard VCSs, where generic line-based differencing algorithms are used, differencing and conflict detection algorithms relying on the graph-based nature of models (e.g., [4,5,6]) are incorporated. They, however, usually do not take semantic information into account. Only Cicchetti et al. [7] present a method for providing semantic awareness for the conflict detection phase by leveraging conflict detection through the adoption of design-oriented descriptions, e.g., design patterns, endowed with custom conflict specifications. In the area of ontology engineering, SemVersion [8] performs semantic difference calculations on the basis of the semantics of RDF and in the field of software engineering, the systems MohadoRef [9] and the approach by Ekman and Asklund [10] use knowledge about the occurrence and the meaning of the refactorings.

Conflict Resolution. In most systems user support is usually restricted to the bare visualization of the conflicts. For example, on model check-in, CoObRa [11] takes the most recent version of the model within the repository, and based on the change protocol, the new changes are stepwise incorporated and presented to the user. Oda and Saeki [12] propose a generator for modeling editors which directly include versioning capabilities. Odyssey-VCS [13] can show the differences either directly in the modeling tool or in a standalone client based on a tree like representation. Only MolhadoRef offers concrete resolution suggestions to users when conflicts have been detected.

Adaption. Only the minority of VCSs provide adaptation mechanisms for the conflict detection and resolution phase. Odysee-VCS allows the configurability of the unit of versioning. In the approach of Cicchetti et al. flexibility can be achieved by the introduction of new and adaption of existing weaving models. The systems CoObRa and Odyssey-VCS provide explicit interfaces for the integration in arbitrary modeling environments.

Summarizing, none of the VCSs covers all issues identified in Section 2 in order to realize appropriate model versioning.

4 AMOR – Adaptable Model Versioning

In this section, the vision of *AMOR (Adaptable Model Versioning)* is proposed for tackling the aforementioned issues and the conceptual architecture for realizing AMOR is presented.

4.1 Goals of AMOR

The main goal of AMOR is the support of mechanisms to leverage version control techniques in the area of MDE. AMOR pursues three goals, each of them leading to several research challenges, as described in the following.

The first goal of AMOR is to achieve **precise conflict detection**, i.e., previously undetected conflicts and previously wrongly indicated conflicts should be avoided. To achieve this, AMOR considers knowledge about the type of modifications the models have undergone in the course of their evolution and knowledge about the semantics of the modeling concepts, leading to the following two main challenges:

Semantic-based Conflict Detection. One of the challenges is to understand the semantics of a DSML in a way that the detection of additional conflicts and the elimination of wrongly reported conflicts are facilitated. Thereby, representations are necessary which (1) explicate the *static* as well as the *behavioral semantics*, i.e., mapping the concepts to a representation that is able to point out specific aspects like in case of behavioral semantics the flow of control [14,15], and which (2) also deals with *semantically equivalent concepts*, i.e., mapping equivalent concepts to a common subset eliminating syntactic sugar of modeling languages.

Operation-based Conflict Detection. The second challenge includes the problem of exploiting the knowledge about the executed operations during model evolution. The knowledge, for example, that a modeler has applied some kind of refactoring pattern, which inherently indicates the modeler's intention behind a modification, is more meaningful for the conflict detection phase than the knowledge that a modeler has applied some unrelated set of basic insert, update and delete operations.

The second goal of AMOR is to provide means for **intelligent conflict resolution support**, specifically aiming at techniques for the representation of differences between model versions and relieving users from repetitive tasks by suggesting proper resolution strategies, thus enhancing productivity and consistency of versioning. This second goal leads to the following two main challenges:

Graphical Visualization of Differences. The first basic challenge is to find an appropriate graphical representation of conflicting modifications in order to alleviate the perception thereof either in the form of (1) the models' concrete syntax or, if not applicable, in the form of (2) the models' abstract syntax.

Suggestions for Conflict Resolutions. The second, and more demanding, challenge is to provide valuable suggestions in the conflict resolution phase stemming either from explicit cooperate design principles (i.e., *static knowledge*) or from implicit recurring application of best practices (i.e., *dynamic knowledge*), which can be made available by observing the user behavior in the conflict resolution phase. This comprises necessary solutions for (1) automatically discovering meaningful conflict resolution patterns with appropriate data mining techniques, (2) establishing similarity measures between resolution situations in order to match a current situation with a pattern stored in the repository, and finally, (3) developing a storage format capturing knowledge about the users' resolution behavior and conflict resolution patterns.

The third goal of AMOR is to provide an **adaptable versioning framework** allowing for proper versioning support while ensuring generic applicability for various DSMLs. That means the AMOR framework can be used in two different ways, either in a generic sense, i.e., out of the box, or by adapting the framework to DSMLs and their corresponding modeling tools on the basis of certain well-defined extension points. With this, the user is empowered to flexibly balance between reasonable adaptation efforts and the needed level for versioning support. The challenges associated with this research goal comprise several kinds of adaptations, partly based on the goals described above:

Adaptation of Conflict Detection. The AMOR framework will be designed in order to allow incorporation of operation-based and semantic-based mechanisms as described above additionally to basic state-based versioning mechanisms requiring only the model versions. Furthermore, adaptation of the versioning granularity should be possible, regarding the semantics of the modeling concepts provided by the DSML (e.g., attributes, classes or whole packages).

Adaptation of Conflict Resolution. The AMOR framework allows adaptation of the level of conflict resolution support, optionally incorporating static and/or dynamic conflict resolution knowledge.

Adaptation of Tool Integration. Finally, the integration of the AMOR framework into the user's modeling tool requires flexibility to allow for a tight coupling in terms of a plug-in mechanism or a loose integration requiring a separate versioning front-end. For exploiting the full potential of AMOR's capabilities, a tight integration is desirable which is unfortunately not possible with all modeling tools.

4.2 Conceptual Architecture of AMOR

In the following, we elaborate on the conceptual architecture of the AMOR framework for providing the above described model versioning support. As can be seen in Figure 1, the AMOR framework is divided into a *front-end* part (i.e., a client) and a *back-end* part (i.e., a repository with versioning functionalities) as known from existing VCSs. The front-end part consists of a so-called *Versioning Assistant* for allowing the modeler to interact with the back-end, which is either a generic stand-alone application for all types of models based on OMG's MOF standard or a specialized plug-in developed for a specific modeling tool (cf. left side of Figure 1). Both have to support the modeler at the check-in of models into the model repository. Thereby, the most challenging task for the modeler is the resolution of (potential) conflicts between model versions.

The back-end part supports the overall versioning process with its subtasks *model comparison*, *conflict detection*, *conflict resolution*, and finally, *model merging* (cf. right side of Figure 1). For this, AMOR initially provides generic state-based versioning, which can be used out-of-the-box, independent of specific modeling languages or tools. However, for improving conflict detection and conflict resolution, AMOR comprises two dedicated subcomponents for the adaptation of the generic versioning capabilities (cf. *Advanced Conflict Detector* and *Conflict Resolution Reasoner* in the middle of Figure 1). First, the *Advanced Conflict Detector* may track editing operations from a modeling tool in order to utilize the model comparison and conflict detection phases for more precise conflict detection reports. Second, an AMOR administrator is able to explicitly define semantic issues for used DSMLs. In contrast to a conflict detection purely based on the structural comparison of models, this helps to detect additional, undetected conflicts and to avoid falsely indicated ones. Moreover, the *Conflict Resolution Reasoner* allows an AMOR administrator to define a set of initial conflict resolution patterns which may be applied by modelers in the conflict resolution phase using the *Versioning Assistant*. The *Conflict Resolution*

Reasoner is not only adaptable by the AMOR administrator, but provides means for mining additional conflict resolution patterns by observing and analyzing resolution operations performed by the modelers.

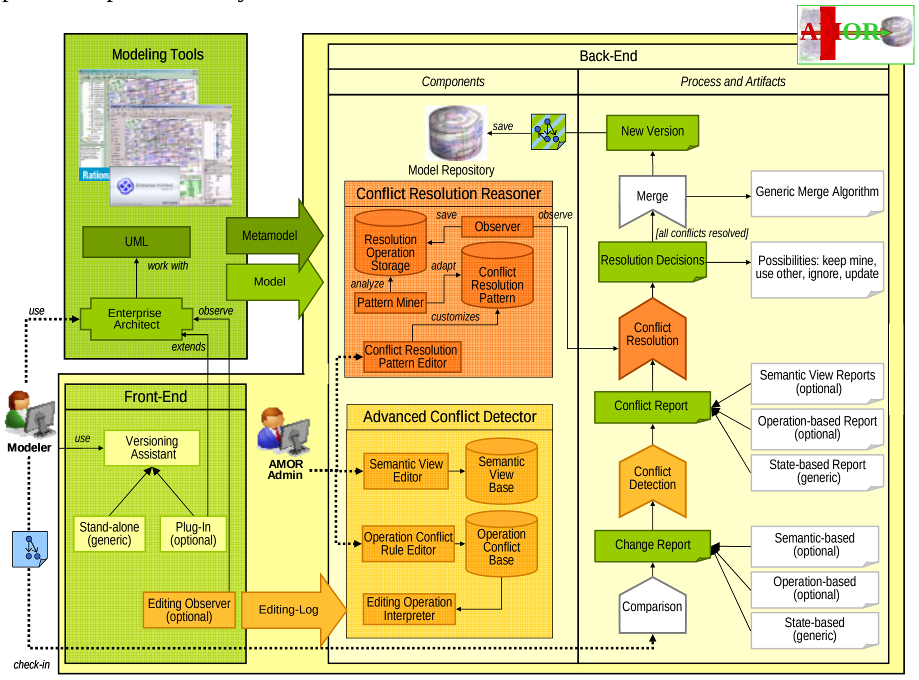

Figure 1: Conceptual Architecture of AMOR

5 Prototypical Implementation

Implementation of AMOR. For demonstrating the feasibility of our proposed approach, a prototypical model versioning system is being implemented based on Eclipse open source technologies. A fully generic *Versioning Assistant* is developed for visualizing conflicts between model versions and functionalities for their resolution. As a model repository, we use the Eclipse Modeling Framework (EMF) on which we base our generic model versioning framework. Subsequently, the *Advanced Conflict Finder* and the *Conflict Resolution Reasoner* subcomponents are placed on top of this generic framework. The *Advanced Conflict Finder* will provide a graphical editor for defining semantic views [15] – a promising approach for semantically enhancing versioning systems proposed in the course of our previous research. A first implementation of a VCS incorporating semantics for conflict detection by means of semantic view definitions called SMoVer [14,15] already exists. For defining operation-based conflicts, a rule-based textual DSML is necessary. The *Conflict Resolution Reasoner* will yield resolution observations that will be processed by the open source platform Pentaho (http://www.pentaho.com), which provides powerful analysis capabilities in the sense of data mining, thus building the basis for our pattern miner component. Found conflict resolution patterns, in turn, will be stored in a repository. Besides the automatic detection of patterns, also the manual definition thereof should be enabled. Therefore, a graphical *Conflict Resolution Pattern Editor* will be established.

Evaluation of AMOR. The evaluation of AMOR will be achieved in three independent steps. First, the conflict detection enhancements will be quantitatively analyzed in the course of experiments. Second, the conflict resolution phase will be examined in empirical studies. And finally, the whole AMOR system will be adapted for the Enterprise Architect modeling tool which is done in cooperation with Sparx Systems (http://www.sparxsystems.com.au).

Customization of AMOR for Enterprise Architect. In order to demonstrate the integration and adaptation of AMOR with respect to state-of-the-art modeling tools, a case study is planed in cooperation with Sparx Systems in which language-specific conflict detection and resolution for UML as well as dedicated versioning front-ends for Enterprise Architect are developed.

References

1. France, R., Rumpe, B.: Model-driven Development of Complex Software: A Research Roadmap. In: Proc. of the Int. Conference on Software Engineering. IEEE. (2007) 37-54
2. Ximbiot: Concurrent Versions System. http://www.cvshome.org (2008)
3. Tigris.org: Subversion. http://subversion.tigris.org (2008)
4. Treude, C., Berlik, S., Wenzel, S., Kelter, U.: Difference Computation of Large Models. In: Proc. of European Software Engineering Conference. ACM. (2007) 295-304
5. Alanen, M., Porres, I.: Difference and Union of Models. In: Proc. of UML 2003 – The Unified Modeling Languages. Springer, LNCS 2863. (2003) 2-17
6. Rivera, J.E., Vallecillo, A.: Representing and Operating with Model Differences. In: Proc. Objects, Components, Models and Patterns: 46th Int. Conf. TOOLS EUROPE. Springer, PNBIP 11. (2008) 141-160
7. Cicchetti, A., Rossini, A.: Weaving Models in Conflict Detection Specifications. In: Proc. of the ACM Symposium on Applied Computing. ACM Press (2007) 1035-1036
8. Völkel, M.: D2.3.3.v2 SemVersion – Versioning RDF and Ontologies. http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D2.3.3v2.pdf (2008)
9. Dig, D., Nguyen, T., Manzoor, K., Johnson, R.: MolhadoRef: A Refactoring-aware Software Configuration Management Tool. In: Proc. of the Annual ACM Conf. on Object-Oriented Programming, Systems, Languages, and Applications. (2006) 321-335
10. Ekman, T., Asklund, U.: Refactoring-aware Versioning in Eclipse. Electronic Notes in Theoretical Computer Science 107, Elsevier. (2004) 57-69
11. Schneider, C., Zündorf, A.: Experiences in using Optimistic Locking in Fujaba (Position Paper). In: Proc. of the Workshop on Comparison and Versioning of UML Models. (2007)
12. Oda, T., Saeki, M.: Generative Technique of Version Control Systems for Software Diagrams. In: Proc. of the IEEE Int. Conference on Software Maintenance. (2005)
13. Murta, L., Corrêa, C., Prudêncio, J.G., Werner, C.: Towards Odyssey-VCS 2: Improvements over a UML-based Version Control System. In: Proc. of the Int. Workshop on Comparison and Versioning of Software Models. ACM. (2008) 25-30
14. Altmanninger, K.: Models in Conflict – Towards a Semantically Enhanced Version Control System for Models. In: Proc. of Models in Software Engineering. Springer, LNCS 5002. (2008) 293-304
15. Altmanninger, K., Bergmayr, A., Kotsis, G., Schwinger, W.: Semantically Enhanced Conflict Detection between Model Versions in SMoVer by Example. In: Proc. of the Int. Workshop on Semantic-Based Software Development. (2007)