A framework for collaboratively editing domain specific models

Koshima, Amanuel; Englebert, Vincent; Thiran, Philippe

Published in:
Doctoral Symposium of the 25th European Conference on Object-Oriented Programming

Publication date:
2011

Document Version
Early version, also known as pre-print

A Framework for Collaboratively Editing Domain Specific Models

Amanuel Koshima
Vincent Englebert (promoter) and Philippe Thiran (co-promoter)

PReCISE Research Center, University of Namur, Belgium
{amanuel.koshima, vincent.englebert, philippe.thiran}@fundp.ac.be

Abstract. Modeling process most of the time demands collaboration among a number of engineers so that DSML tools need to be support collaboration. In this work, we will present DiCoMEF that will be developed to support collaborative modeling.

Keywords: Collaborative Work, Model Versioning, DSM

1 Problem Description

Model Driven Engineering (MDE) is a software engineering technique aiming to raise the level of abstraction of a software development from code to model [5]. MDE uses Domain Specific Modeling Languages (DSML) to specify the structure, behavior and requirements of the applications within specific domains [5]. DSML describes the business concepts at different abstraction levels using model, meta-model and meta-meta-model. A model is an abstraction of a software system. A meta-model is a language that defines a valid instances models. Likewise, models are specified by a meta-model, meta-models are described by meta-meta-models (i.e. EMF/Ecore [12]).

Like other software artifacts, meta-models could also evolve during software development life cycles [6]. As a result, the existing models may no longer conform to the new version of meta-model. Therefore, these models need to be co-evolved in order to conform to the adapted meta-model.

Although most of the DSM tools developed in the past consider the software modeling process as a single user task, it usually requires collaboration among a large number of engineers with different specialties [4]. Hence, there is a need for group members to share models and meta-models and synchronize their activities. Shared modeling artifacts could be edited concurrently by different users without prior consultation. As a result, these different versions of artifacts might be in conflict. The management of inconsistencies is the main challenge for the implementation of collaborative model editing framework.

A central repository with locks and merge mechanisms is the most commonly used approach to handle conflicts and ensure collaboration [10]. But, locking technique becomes inadequate when a number of users is getting large [3]. In addition, this approach restricts users to be dependent on a central repository. It

could also introduces administration of access rights which might be cumbersome and causes users dissatisfaction. Other modes of collaboration is where each member of a group has his/her own local copy (meta)model and communicate his/her activities by sending messages [10]. This type of collaboration gives users control over their data and let them work in isolation. It also addresses the problem of being dependent on a single repository. But, it is challenging to keep all copies of (meta)models that could be concurrently modified.

2 Goal Statement

Saeki [11] introduced the use of versioning system to control and manage models and meta-models, which evolve independently. The author did not consider collaboration in his work. EMFStore is a model repository for collaboratively editing EMF models which is implemented based on the premise of on a central repository with copy-merge techniques [9]. MetaEdit+ [8] implements *Smart Mode Access Restricting Technology* (Smart Locks ©) to support concurrent access of shared modeling artifacts that are stored centrally. As it was stated above, this type of collaboration limits developers to work on one central repository.

In [4], Constantin et. al. proposed a reconciliation framework for collaborative model editing. In their work, they suggested a weakly coupled mode of collaboration, where (meta)models are managed in distributed fashion. But, they only provide a theoretical reconciliation framework to support collaborative work without providing a solution. In another work, Mougenot et. al. [10] developed a peer-to-peer collaborative model editing framework called D-Praxis. In Mougenot et. al. proposal, developers exchange sequences of operations used to adapt a model. This approach implemented automatic conflict resolution based on delete semantics and Lamport's clock. Nevertheless, this approach suffers from similar problem of "lost-update". Furthermore, it does not take into consideration the co-evolution of models with their meta-models.

In this context, our research will aim at developing a Distributed Collaborative Model Editing Framework (DiCoMEF) for modeling tools that are based on EMF/Ecore meta-meta-model definition. In DiCoMEF, both models and meta-models could be freely distributed without having constraints like central repository. Engineers will be able to carry out their work independently without prior consultation with other users of the same model. Besides, sequences of operations that are used to adapt models and meta-models are captured and used as a means to propagate local modifications from one developer to others. These sequences of operations are also used to detect conflicts and merge two versions of the same meta-model into one. Indeed, the reconciliation of conflicting modifications is supervised by a controller (human agent).

In this work, we use COPE [7] to record edit operations performed on meta-model and to generate and attach instructions of how to migrate models. We adopted COPE because it is industrially tested and it has tool support [1]. But, COPE does not support collaboration. So that we will develop a tool which provides import and export facilities with conflict detection and merging for

concurrently edited meta-models. We will also develop a tool to record edit operations of models and integrate with import and export facilities. In addition, a facility for engineers to annotate edit operations with multimedia files to express their rationale of changes will be provided.

3 Method

Collaboration Framework: DiCoMEF is a distributed collaborative system in which every engineer has his/her own local copy of (meta)models and edit them locally. Changes concern (meta)models are captured and used to propagate local modifications to other engineers. Besides, engineers could also annotate the rationale of changes they make with multi-media files. As part of change management, a controller is responsible for supervise modifications of (meta)models. A controller detects conflicts and reconciles them under the supervision of the engineer who proposed changes. Afterwards, s/he propagates accepted modifications to other engineers.

Model Comparison compares two (meta)models so as to drive their differences. *State-based comparison* and *change-base comparison* are the most commonly used approaches [3]. The first approach compares the information of (meta)models whereas the second one compares the log of changes recorded from the latest common version. Unlike *state-based comparison*, *change-base comparison* captures the time sequence of changes that is important to understand changes, conflict detection, and merging [3]. Besides, it keeps composite operations in the same context [3]. *State-based comparison* is computationally expensive. In this work, we opted for the *change-base comparison.*

Conflict Detection and Resolution: according to [3], conflicts are a set of operations that causes inconsistency. These conflicts could be *textual, syntactic, composite, or semantic* conflicts. Conflicts might be solved manually, semi-automatically or automatically. Manual conflict detection is time consuming and error prone to deal with complex (meta)models. Automatic conflict resolution is not applicable in most situations, because conflict detection and resolution is domain specific. This work adopts a semi-automatic conflict detection approach.

Model Merging: it is a process of integrating concurrently edited (meta)models, which have the same base version into a new (meta)model. The most commonly used merging techniques are *raw merge, two-way merge and three-way* merge [3]. The last one gives the better result as compared to the other two approaches. So that we adopted a three-way mergining technique in our work.

Model Migration : it is a process of adapting models in response to meta-model evolution in order to keep the existing models conform to a new version of meta-model [7]. *Difference-based approaches* and *operation-based approaches* are commonly adopted approach for model migration. *Difference-based approach* could not always derive correct migration instructions. *Operation-based approaches* derives migration instructions from sequences of operations that adapt a meta-model. This approach keeps composite operations in the same context, which could help to understand the intention of a user.

Change Propagation: engineers can communicate their works in peer-to-peer communication mode, but it is difficult to keep all local copies consistent that are concurrently edited by different engineers. Another mode of change propagation is where every developer sends sequences of changes to a controller and then the controller supervises modifications and propagates accepted changes to other members. We adopted the second mode of communication and will investigate different policies on how to apply propagated changes on all local copies.

Validation: DiCoMEF will be tested and evaluated with USiXML project. USiXML is an UML/XML/ECore based meta-model which offers a common way to specify User Interface (UI) independent of the underlying platform and programming languages [2] .

References

1. COPE, coupled evolution of metamodels and models. `http://cope.in.tum.de/pmwiki.php`, 2011.
2. UsiXML, user interface extensible markup language. `http://www.usixml.org/`, 2011.
3. K. Altmanninger, M. Seidl, and M. Wimmer. A survey on model versioning approaches. *IJWIS*, 5(3):271–304, 2009.
4. C. Constantin, V. Englebert, and P. Thiran. A reconciliation framework to support cooperative work with DSM. In *Proceedings of the First International Workshop on Domain Engineering held in conjunction with CAiSE'09 Conference, collection CEUR-WS.org*, volume 457, 2009.
5. J.-M. Favre. Towards a basic theory to model model driven engineering. In *In Workshop on Software Model Engineering, WISME 2004, joint event with UML2004*, 2004.
6. B. Gruschko. Towards synchronizing models with evolving metamodels. In *In Proc. Int. Workshop on Model-Driven Software Evolution held with the ECSMR*, 2007.
7. M. Herrmannsdoerfer, S. Benz, and E. Juergens. COPE: A Language for the Coupled Evolution of Metamodels and Models . In *Proc. of the 1st International Workshop on Model Co-Evolution and Consistency Management.* ACM, 2008.
8. S. Kelly. Case tool support for co-operative work in information system design. In C. Rolland, Y. Chen, and M. Fang, editors, *Information Systems in the WWW Environment*, volume 115 of *IFIP Conference Proceedings*, pages 49–69. Chapman & Hall, 1998.
9. M. Koegel and J. Helming. EMFStore: a model repository for emf models. In J. Kramer, J. Bishop, P. T. Devanbu, and S. Uchitel, editors, *ICSE (2)*, pages 307–308. ACM, 2010.
10. A. Mougenot, X. Blanc, and M.-P. Gervais. D-praxis: A peer-to-peer collaborative model editing framework. In *Proceedings of the 9th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, DAIS '09, pages 16–29, Berlin, Heidelberg, 2009. Springer-Verlag.
11. M. Saeki. Configuration management in a method engineering context. In E. Dubois and K. Pohl, editors, *CAiSE*, volume 4001 of *Lecture Notes in Computer Science*, pages 384–398. Springer, 2006.
12. D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework 2.0.* Addison-Wesley Professional, 2nd edition, 2009.