Model Co-evolution and Consistency Management (MCCM'08)

Dirk Deridder[1], Jeff Gray[2], Alfonso Pierantonio[3], and Pierre-Yves Schobbens[4]

[1] Vrije Universiteit Brussel, Belgium, dirk.deridder@vub.ac.be
[2] University of Alabama at Birmingham, USA, gray@cis.uab.edu
[3] Università degli Studi dell'Aquila, Italy, alfonso@di.univaq.it
[4] Université de Namur, Belgium pierre-yves.schobbens@fundp.ac.be

Abstract. The goal of the workshop was to exchange ideas and experiences related to Model (Co-)evolution and Consistency Management (MCCM) in the context of Model-Driven Engineering (MDE). Contemporary MDE practices typically include the manipulation and transformation of a large and heterogeneous set of models. This heterogeneity exhibits itself in different guises ranging from notational differences to semantic content-wise variations. These differences need to be carefully managed in order to arrive at a consistent specification that is adaptable to change. This requires a dedicated activity in the development process and a rigourous adoption of techniques such as model differencing, model comparison, model refactoring, model (in)consistency management, model versioning, and model merging. The workshop invited submissions from both academia and industry on these topics, as well as experience reports on the effective management of models, metamodels, and model transformations. We selected ten high-quality contributions out of which we included two as best-papers in the workshop reader. As a result of the high number of participants and the nice mix of backgrounds we were able to debate lively over a number of pertinent questions that challenge our field.

1 Introduction

In general, software artifacts and applications are subject to many kinds of changes. These range from technical changes due to rapidly evolving technology platforms, to modifications resulting from the natural evolution of the business that is supported. This includes changes at all levels, from requirements through architecture and design, to source code, documentation and test suites. They typically affect various kinds of models including data models, behavioral models, domain models, source code models, goal models, etc. Coping with and managing the changes that accompany the evolution of software assets is therefore an essential aspect of Software Engineering as a discipline.

Model-Driven Engineering (MDE) is an approach to software design and development in which models are the primary artifacts of software development. The major objective of MDE is to increase productivity and reduce time-to-market by raising the level of abstraction. In part this is done by using concepts

closer to the problem domain instead of those offered by programming languages. Such models represent domain-specific concepts and conform to metamodels.

A core task of MDE is the manipulation and transformation of models. Thus, Model (Co-)evolution and Consistency Management (MCCM) are crucial activities to cope with the natural changes of the corresponding software system. Currently, there is an increasing need for more disciplined techniques and engineering tools to support a wide range of model evolution activities, including model differencing, model comparison, model refactoring, model inconsistency management, model versioning and model merging.

Recently, a number of works devoted to the detection of differences between models has emerged to foster enhanced model management practices. The exploitation of differences is an appropriate solution for version management, because in general the complete system model is far larger than the modifications that occur from one version to another. Apart from these works, further research is required to address the rest of the model evolution activities (refactoring, inconsistency management, versioning, etc.). Moreover, the different dimensions of evolution make the problem intrinsically difficult because modifications can reflect coherent adaptations of correlated artifacts at several layers of the metamodeling architecture. For example, some well-formedness rules can be invalidated when a metamodel evolves. The same happens with the associated model transformations. Furthermore, model adaptations should be propagated to artifacts interconnected by means of model transformations. Finally, the evolution of model transformations should be reflected in both source and target models.

In addition, there is a substantial difference between the modeling of evolution and the evolution of models. There are plenty of works on the former topic, while our proposed workshop focuses on the evolution of models. One of the goals of this workshop was to explain and clarify the difference between these two notions, by explicitly identifying the concepts and mechanisms involved in each one. In particular, we targeted the cross-fertilization of both the MDE and software evolution communities. This is why we considered models in a very broad sense to allow researchers from different communities to identify and discuss commonalities/differences among their specific MCCM problems.

2 About the Workshop

The workshop was a great success[5]. We had 59 registered participants which resulted in many interesting and lively discussions. In total we accepted 10 submissions for presentation which are listed below (presenters are underlined). The contributions marked with an **(*)** were selected as best-papers and are included in the Workshop Reader. The other papers are available in the electronic workshop proceedings[6].

[5] MCCM'08 was organised in cooperation with the MoVES network, funded by the Belgian State, Belgian Science Policy http://moves.vub.ac.be/.

[6] MCCM'08 Workshop Proceedings: http://www.info.fundp.ac.be/mccm/

An Inconsistency Handling Process,
by Ragnhild Van Der Straeten

Retainment Rules for Model Transformations,
by Thomas Goldschmidt and Axel Uhl

(*) Triple Graph Grammars or Triple Graph Transformation Systems? A Case Study from Software Configuration Management,
by Bernhard Westfechtel, Thomas Buchmann and Alexander Dotor

MOD2-SCM: Experiences with Co-evolving Models when Designing a Modular SCM System,
by Thomas Buchmann, Alexander Dotor and Bernhard Westfechtel

Efficient Recognition and Detection of Finite Satisfiability Problems in UML Class Diagrams: Handling Constrained Generalization Sets, Qualifiers and Association Class Constraints,
by Azzam Maraee, Victor Makarenkov Makarenkov and Mira Balaban

COPE: A Language for the Coupled Evolution of Metamodels and Models,
by Markus Herrmannsdoerfer, Sebastian Benz and Elmar Juergens

(*) On Integrating OCL and Triple Graph Grammars,
by Duc-Hanh Dang and Martin Gogolla

Model Engineering using Multimodeling,
by Christopher Brooks, Chih-Hong Cheng, Thomas Huining Feng, Edward A. Lee and Reinhard von Hanxleden

AMOR Towards Adaptable Model Versioning,
by Kerstin Altmanninger, Gerti Kappel, Angelika Kusel, Werner Retschitzegger, Martina Seidl, Wieland Schwinger and Manuel Wimmer

Co-Evolution and Consistency in Workflow-based Applications,
by Mario Sanchez, Jorge Villalobos and Dirk Deridder

After the presentations we scheduled a number of plenary discussions. Each discussion was focused on a particular question that addressed a major theme in model consistency management and model (co-)evolution.

What are the steps required to install a full-fledged MCCM process?
The need for having a rigourous consistency and co-evolution process in place was widely acknowledged. A number of participants pointed out that setting up and (technologically) supporting an MDE process is already a big challenge. It was agreed upon that more effort should go into defining an integral approach.

What are the MCCM challenges when managing large models?
Several issues were discussed related to working with large models (both in terms of size and diversity). In particular, the scalability of existing consistency handlers was questioned, this not only with respect to the computational side but also to the possible cascade of inconsistencies. A challenge in the future will be to derive which inconsistencies to address first in order to reduce the size of problems reported. In addition, it is desirable to guide developers to cope with the 'unstructured' set of inconsistencies (e.g., by

tagging the inconsistencies with their type and importance). Also, the need to temporarily tolerate inconsistencies was identified, which requires a mechanism to reason with inconsistent models (e.g., by using a kind of 'closest semantic match' to overcome problems).

How does bridging semantic domains impact MCCM?
The main issue discussed was how inconsistency handling techniques are challenged by having to address model elements that stem from different semantic universes. It was suggested that we also need dedicated formalisms to specify cross-domain incompatibilities. This is in line with the growing tendency towards domain-specific modeling languages, and it requires to balance the benefits of domain-independent support versus dedicated support (e.g., dedicated detection engines, formalisms, resolution strategies).

What are the different perspectives on MCCM and how can we learn from each other?
One of the goals of the workshop was to assemble researchers from diverse fields in which MCCM is being addressed. We believe this goal was met as exemplified by the contributions of the participants. Some participants discussed the topic from a generic MDE point of view, whereas others discussed it specific to a particular application domain (e.g., software configuration management, workflow-based applications). Also, in terms of the proposed techniques we had a good distribution of topics (e.g., retainment rules, triple graph grammars, ocl, coupled evolution of metamodels and models). As a result, we were able to hold a lively debate during which we focused on identifying and relating the commonalities and differences of the different perspectives. There was a general consensus that the community would benefit from bringing together existing categorisations of co-evolution scenarios and (in)consistency types. Additionally, it was suggested to record the differences in terminology since there are often subtle semantic deltas that might cause misunderstandings (e.g., co-evolution versus coupled evolution versus co-adaptation). During the workshop we were already able to clarify a number of misconceptions. The alignment of terminology was also useful to identify how we might possibly benefit from each other's techniques and approaches. Therefore, a number of workshop participants suggested to set up an online community in the form of a wiki. It is our hope that this wiki will further enable the cross-fertilisation of both the MDE and software evolution communities with respect to consistency and (co-)evolution problems.

3 Acknowledgements

We thank our programme committee members for their help in the paper selection process: Jean Bézivin, Rubby Casallas, Antonio Cicchetti, Serge Demeyer, Stéphane Ducasse, Vincent Englebert, Jean-Marie Favre, Tudor Gîrba, Reiko Heckel, Viviane Jonckers, Frederic Jouault, Ralf Lämmel, Kim Mens, Tom Mens, Jonathan Sprinkle, Antonio Vallecillo, Ragnhild Van Der Straeten. We also thank Anthony Cleve and Andreas Classen for the considerable effort they put into the practical organisation of MCCM'08.