A Framework for Handling Variants of Software Models

Christian Pichler*
Research Studios Austria
Thurngasse 8/3/20
1090 Vienna, Austria
cpichler@researchstudio.at
http://www.big.tuwien.ac.at/staff/cpichler.html

ABSTRACT

The United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) envisions seamless information exchange between business partners in electronic commerce. Therefore, UN/CEFACT provides the UML Profile for Core Components for the definition of document models based on UML class diagrams. Having used this approach for three years in practice, it became evident that managing document model versions is a prerequisite for successfully utilizing Core Components. While managing software versions in the area of Software Engineering is well understood and successfully applied in industrial projects, the direct application of the same techniques for versioning models is conditionally appropriate. In this research abstract we propose to combine techniques from traditional Software Configuration Management with the concepts of reference modeling, where similar problems are addressed in a different context.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Software Configuration Management*; D.2.9 [**Software Engineering**]: Management—*Life cycle*

Keywords

Model-Driven Engineering, Reference Modeling, Version Management

1. INTRODUCTION

Software systems are designed for meeting the requirements of a certain domain. However, as time evolves, the requirements typically change, demanding for a corresponding adaption of software systems. For successful Software Engineering it is essential to control the evolution of software systems which is addressed in Software Configuration Management (SCM) [9]. One aspect of SCM is managing different versions of software systems. Versions are differentiated into revisions as well as variants [4]. Revisions emerge along the time dimension and replace preceding revisions whereas variants intentionally coexist. Nowadays, models are an important aspect of Software Engineering with the purpose of documenting software systems or for performing Model-Driven Engineering (MDE) [3]. For effectively utilizing models, such as models created based on UN/CEFACT's UML Profile for Core Components (UPCC) [6], proper management of versions is needed. However, currently the support for managing model versions is limited.

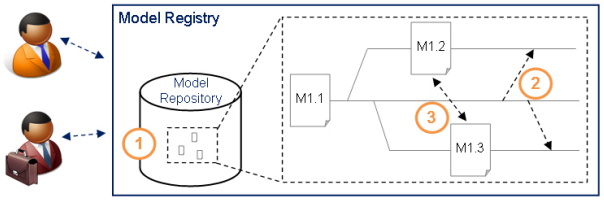

Figure 1: Model Registry.

2. MOTIVATION AND RESEARCH GOALS

Figure 1 illustrates a typical scenario as it would appear in managing model variants. In the following we elaborate on selected problems occuring in managing variants of models created using the UPCC, which are structural models based on UML class diagrams.

First, a repository capable of storing models as well as *managing model variants* is needed (cf. Figure 1, Mark 1). We propose to follow the idea of utilizing reference modeling techniques [10] to provide an alternative approach for managing model variants.

Second, *preserving consistency* across variants of models is addressed. Variants are based on a core model. Assuming that the core model changes, inconsistencies between each of the variants and the core model arise. Suppose that a core model M1.1 has two variants, M1.2 and M1.3. Then one particular element is removed from the core model. In order to preserve consistency, all changes to the core model need to be applied to its variants (cf. Figure 1, Mark 2).

Third, creating variants of models may result in *duplicate variants*. When models evolve into different variants it is often the case that models are, though unintentionally, developed simultaneously. Variants are created based on restricting a core model. As a result different modelers may apply the same restrictions on the core model resulting in duplidate model variants (cf. Figure 1, Mark 3).

*PhD Student at the Vienna University of Technology

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICSE '10, May 2-8 2010, Cape Town, South Africa
Copyright 2010 ACM 978-1-60558-719-6/10/05 ...$10.00.

Overall, we propose a model registry (cf. Figure 1) capable of managing model variants. In particular, the benefits of such a registry include a repository for storing models and their variants, proper concepts for preserving consistency across model variants, as well as detecting similar model variants in order to reduce model complexity and duplication. Furthermore, another project in our group deals with model versioning with respect to revisions [1]. Therefore, the proposed approach would complement the research work for managing model revisions. Furthermore, the combination of both approaches serves as a basis for collaborative development of models.

3. RELATED WORK

In the following we describe related work which has already been analyzed as part of the literature study that we are currently working on. Due to space limitations we provide representatitive examples. For versioning in the area of Software Engineering a number of approaches exist. An extensive overview is provided in [4]. Furthermore, for managing variants, feature models [5] as well as software product lines [8] are used. An overview of current efforts in managing models versions is provided in [2]. Furthermore, in [10], a profound overview on reference modeling, also focusing on the different techniques of reference modeling, is provided. [7] presents operators for finding correspondences between models relevant for detecting duplicate model variants.

4. RESEARCH QUESTIONS

For addressing the problems stated earlier the following set of research questions is identified.

Variant Management. The first research question addresses defining proper concepts for a model repository capable of storing models and their variants. In particular, two conceptually different approaches are chosen and consolidated. The first approach is motivated by managing variants in Software Engineering such as feature models or software product lines. The second approach exploits the use of reference modeling. At this stage the reference modeling technique Configuration seems promising.

Change Propagation. First, pre-conditions for changes applied to models and their variants are determined. Second, concepts for detecting differences between models and their variants are defined where different aspects, such as the granularity of changes, are addressed. Third, concepts for propagating changes to model variants are established. Such concepts allow to preserve consistency across models and their variants.

Similarities. The third research question deals with defining concepts allowing to detect similar model variants. For detecting similarity of model variants, structure as well as semantics should be considered. Having such concepts at hand, reusing models is supported resulting in the reduced complexity of models.

5. RESEARCH METHODS

In order to address the previously identified research questions we follow the conceptual and constructive methodological approach. In particular, the following steps are performed in the order listed.

Literature Study and Evaluating Systems. The first step that we are currently working on is studying literature in the area of model versioning. The purpose of literature study is to identify existing concepts for managing variants. Furthermore, to the best of our knowledge there is no survey on tools which support managing model variants. Therefore, as a first step, it is planned to conduct a survey on tool support for managing model variants.

Architectural Design and Prototype Implementation. Based on an overview of existing work as well as existing tools it is possible to propose an architecture for a prototype. Besides the architectural design, proper technology for implementation is chosen. One requirement that the technology chosen must fulfill is that it should be compatible with today's state-of-the-art modeling environments. For preparing the evaluation of the research conducted a prototype is implemented.

Evaluation. In order to prove the result of the research conducted we evaluate the concepts with our prototype based on a set of case studies. Since we actively participate in UN/CEFACT, we have access to a pool of models which will be used in our case studies. One candidate for evaluation are UN/CEFACT's Core Components which allow conceptual modeling of business documents.

6. REFERENCES

[1] K. Altmanninger, G. Kappel, A. Kusel, W. Retschitzegger, M. Seidl, W. Schwinger, and M. Wimmer. AMOR - Towards Adaptable Model Versioning. In *Proceedings of the 1st International Workshop on Model Co-Evolution and Consistency Management*, 2008.

[2] K. Altmanninger, M. Seidl, and M. Wimmer. A Survey on Model Versioning Approaches. *International Journal of Web Information Systems*, 5(3):271–304, 2009.

[3] J. Bézivin. On the Unification Power of Models. *Software and System Modeling*, 4(2):171–188, 2005.

[4] R. Conradi and B. Westfechtel. Version Models for Software Configuration Management. *ACM Computing Surveys*, 30(2):232–282, 1998.

[5] K. Czarnecki, S. Helsen, and U. W. Eisenecker. Formalizing Cardinality-based Feature Models and their Specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.

[6] C. Huemer, P. Liegl, and C. Pichler. A registry model for UN/CEFACT's Core Components. In *Proceedings of IEEE International Conference on Service-Oriented Computing and Applications*. IEEE, 2009.

[7] S. Nejati, M. Sabetzadeh, M. Chechik, S. M. Easterbrook, and P. Zave. Matching and Merging of Statecharts Specifications. In *Proceedings of the International Conference on Software Engineering*, pages 54–64. IEEE, 2007.

[8] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer, 2005.

[9] W. F. Tichy. Tools for Software Configuration Management. In *Proceedings of the International Workshop on Software Version and Configuration Control*, pages 1–20, 1988.

[10] J. vom Brocke. Design Principles for Reference Modeling. In *Reference Modeling for Business System Analysis*. Idea Group Publishing, 2006.