

Towards Structured Revisions of Meta Models and Semi-Automatic Model Migration

Boris Gruschko¹
boris.gruschko@sap.com

SAP Research, CEC Karlsruhe, 76131 Karlsruhe, Germany

Abstract. Model Driven Software Development (MDS) is increasingly gaining attention in the software development community. The success of the MDS approach depends on the usage of artifact managing tools, due to the number of different artifacts produced during MDS based project's life-cycle. However, the available tools and methods concentrate on model authoring. This leads to the negligence of software logistic problems introduced by MDS. This paper outlines the problem of Meta Model versioning and Model migration in scope of enterprise scale applications.

1 Introduction

Models are central artifacts of MDS based projects. A Model is described in terms of a Meta Model, which represents the vocabulary of the modeling domain. A Meta Model conforms to the Meta Meta Model, which is abstract enough to allow for Meta Model authoring in envisioned development domains. For the rest of this paper, these modeling levels will be referred to as $M1$ through $M3$ correspondingly. This terminology has been proposed by the MOF specification[1].

The $M3$ models are agnostic to the pursued development domain. Therefore, they are considered to be unchangeable in scope of the software project life-cycle. $M3$ models are normally hardwired into the modeling infrastructure, used as substrate for modeling tools. Most known $M3$ models are Ecore[2] and MOF[1, 3].

In contrast to $M3$ models, $M2$ and $M1$ models are subject to modification during project progression. Changes to $M1$ models are being introduced in course of day to day work on the project. The changes of $M2$ models result from insights into modeling domain, won during the project's life-cycle.

The eventual modifications of $M2$ models are arising the problem of $M1$ model consistency. The $M1$ models, created to conform to an existing $M2$ model may become invalid due to compatibility breaking changes introduced to $M2$ models. The invalid $M1$ models have to be adopted to the new versions of $M2$ models to retain their validity.

The rest of this paper is organized as follows: section 2 extends the presented problem definition. Section 3 decomposes the stated problem and provides an outline of research topics relevant to problem resolution. Section 4 presents a

preliminary survey of work related to the presented subject. Finally, section 5 concludes this position paper.

2 Problem Definition

Changes introduced to $M2$ models during project life-cycle can break already existing $M1$ models. The broken $M1$ models would not be consistent in regard to their $M2$ models and therefore not utilizable by the modeling infrastructure. The introduced $M2$ model modifications do not necessarily break the $M1$ models. There are types of modifications, which alter the $M2$ model, while retaining the $M1$ model's consistency.

An example of $M1$ model breaking $M2$ model modification is the narrowing of cardinality bounds on association ends. If for example, the upper cardinality bound is lowered below the cardinality number of the corresponding link set in the $M1$ model, the $M1$ model becomes inconsistent. This inconsistency can not be resolved by automatic means. A human decision on link deletion is required to resolve it. An example of automatically resolvable $M1$ model breaks is the renaming of $M2$ model's elements. Because the $M2$ model elements are assigned a unique identifier, the changes of $M2$ model element names can be tracked and propagated to $M1$ models automatically. Changes which do not break the $M1$ model, are model extensions. For example an addition of a class to the $M2$ model will not break the $M1$ models, unless association ends with lower cardinality bound larger than zero are pointing to this class.

The breaking of $M1$ models, due to $M2$ model modifications, poses a significant threat to MDS adoption in enterprise application development. Enterprise Resource Planning(ERP) systems require to be maintained and upgraded for decades. Further, these systems are subject to frequent modifications, due to changing business needs of the enterprise. Naturally, the $M2$ models are also subject to changes, either because of a better understanding of the described phenomena, or due to external influences like regulatory requirements. As has been shown in previous paragraph, even small changes in $M2$ models can lead to breaking of the $M1$ models. Therefore structured and systematic means of $M1$ models adoptions to changes in $M2$ models are required. Otherwise, the costs of manual $M1$ model adaptation can exceed the savings of MDS methods adoption.

3 Research Directions

The resolution of the presented model migration problem, arises three main research directions. The first the creation of the difference model, describing the differences between two $M2$ model versions. The second is the question of serialization of difference information, to be presented to the human developer or transported to the $M1$ model migrator. And finally the third is the approach to the classification of $M2$ model changes and $M1$ model migration.

The difference of two $M2$ model versions can be computed in terms of the $M3$ model. Therefore, the outcome of the differences computation would be a set of expressions about deleted, changed and added $M2$ model elements. However, the algorithms for the computation of this set are to be developed and tested in a real modeling environment. Further, it is to be determined if a single step comparison would deliver the desired results. A single step comparison, would compare two $M2$ model revisions without considering the operations which lead to the creation of the newer revision. Another possibility to be evaluated, is the consideration of the trace of operations performed on the first $M2$ model revision. It is to be evaluated, if a Change Meta Model should be created. A Change Meta Model would link to the revisions of $M2$ model providing information on $M2$ model evolution. If a Change Meta Model is feasible, the differences set would represent the population of that $M2$ model.

The serialization of $M2$ model differences is highly depended on decisions made in the difference creation phase. If a change meta model is being created, then standard model serialization means like XMI[4] can be used. Otherwise a custom serialization format for $M2$ model differences has to be developed. Another research topic, is the human readable notation for $M2$ model differences. This notation is needed, to communicate changes between two $M2$ model revisions in human readable format. Difference representations, known from traditional versioning systems like CVS[5], fail to deliver difference serialization in the same notation, as used by the user for $M2$ model authoring. These difference serializations concentrate on textual formats. However, the $M2$ models are normally being defined in a graphical notation. This leads to an undesirable mismatch in authoring and difference notations.

As has been shown in the example in section 2, the migration of $M1$ models between $M2$ model revisions can not be fully automated. Therefore a comprehensible catalog of possible $M2$ model changes has to be created. The emphasis of this catalog is to the classification of $M2$ changes, according to their impact on $M1$ models. Starting with this catalog, algorithms for automatically resolvable breaks resolution can be created. For non-automatically resolvable changes, means for human assistance have to be developed. The main research topic for this direction, is the creation of means, to assist the human migrator in the same concrete syntax, used in the modeling domain. Therefore, the Change $M2$ model to be developed, has to be attached to the concrete syntax, used to create populations of the $M2$ model under consideration. This attachment can for example be expressed in links to a graphical $M2$ model like GMF[6] Graph model. Another problem, posed by model migration, are the changes of model interfaces, introduced by changes of $M2$ models. These changes would inevitably break the tools, built on top of the changed $M2$ model. A solution here, could be the provision of refactoring[7] facilities, migrating the calls to model interfaces in the same manner as $M1$ models are being migrated.

This overview presents three directions, in which the research work in the model migration domain can progress. The topics are by no means finalized. The

research to be performed is expected to reveal major problems in computability and performance domains.

4 Related Work

This section reviews the existing related work on the proposed research topics. This review is by no means complete and is only a preliminary overview, deemed to provide first insights into the general research field.

There are no industrial standards, concerning the topic of *M1* model migration. OMG has published the MOF2 Versioning specification[8]. This standard describes the attachment of versioning information to MOF *M2* models, but excludes the means to obtain the difference information. The *M1* models migration is completely uncovered by this standard.

The general topic has been developed under the term of "Generic Model Management" by Melnik[9, 10]. This work provides the first results in the field of Generic Model Management, as described in [11]. The presented work is still to be evaluated. Firstly, it can provide a basis to build upon, secondly it could be interesting, to embed the proposed research in the general field description of Generic Model Management.

Work on difference algorithms for UML models can be relevant, to the presented topic of differences determination between two versions of a *M2* model. Grischnik[12] provides the algorithm $UMLDiff_{clad}$. The paper by Girschick also provides an overview of related work in this field. Further, the work performed by the database community on schema matching algorithms could be relevant for the difference determination algorithms.

Sprinkle[13] performed work on migration of *M1* models in face of *M2* model changes. This work however concentrates on the preservation of semantic consistency of *M1* models. It is to be evaluated, if the proposed approach is fruitful, when applied to the preservation of structural features of *M1* models to be migrated.

5 Conclusion

The presented position paper, suggests work on *M1* models migration in face of *M2* model changes. While all three presented topics are relevant to the proposed work, their combination promises to provide a basis for the inception of a revision system for modeling artifacts, delivered by the MDSD approach.

This topic is highly relevant, if the MDSD approach is to deliver on it's promise of higher development efficiency. Further, the avoidance of notation mismatch between model authoring and model comparison, promises to provide the access, to the proposed capabilities, to users beyond the software engineering community.

References

1. Object Management Group: Meta Object Facility (MOF) Specification — Version 1.4. (2002)
2. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.: Eclipse Modeling Framework. Addison Wesley Professional (2003)
3. Object Management Group: Meta Object Facility (MOF) 2.0 Core Final Adopted Specification. (2004)
4. Object Management Group Framingham, Massachusetts: OMG XML Metadata Interchange (XMI) Specification Version 2.0. (2003)
5. Cederqvist, P.: Version Management with CVS. Signum Support AB, Box 2044, S-580 02 Linköping, Sweden. (1993)
6. Eclipse Foundation: GMF Home Page. (<http://www.eclipse.org/gmf>)
7. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. Addison Wesley (1999)
8. Object Management Group: (MOF2 Versioning Final Adopted Specification)
9. Melnik, S.: Generic Model Management: Concepts and Algorithms. Volume 2967 of Lecture Notes in Computer Science. Springer (2004)
10. Melnik, S.: Model management: First steps and beyond. In Vossen, G., Leymann, F., Lockemann, P.C., Stucky, W., eds.: BTW. Volume 65 of LNI, GI (2005) 455–464
11. Bernstein, P.A., Haas, L.M., Jarke, M., Rahm, E., Wiederhold, G.: Panel: Is generic metadata management feasible? In Abbadi, A.E., Brodie, M.L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., Whang, K.Y., eds.: VLDB, Morgan Kaufmann (2000) 660–662
12. Girschick, M.: Difference Detection and Visualization in UML Class Diagrams. Technical Report TUD-CS-2006-5, TU Darmstadt (2006)
13. Sprinkle, J.M.: Metamodel Driven Model Migration. PhD thesis, Vanderbilt University (2003)