Changes Classification in M2 Models

Boris Gruschko
SAP Research
CEC Karlsruhe
Vincenz-Priessnitz-Strasse 1
76131 Karlsruhe, Germany
boris.gruschko@sap.com

Abstract: As Model Driven Software Development gains attention, its utilization become more feasible in large scale software projects. The metamodels (M2 models) are important artifacts of the MDD process. These models capture the modeler's understanding of the problem domain. However, the process of problem domain understanding is not necessarily completed, as the M2 models is being delivered to its consumers. Therefore, the changes of M2 models are inherent to the modeling process. Therefore, the M1 models can erode, when not backwards-compatible changes are being introduced to M2 models. The problem of M1 model erosion in face of M2 model changes becomes a definitive one, when MDD is applied in large scale development projects. We therefore propose a structured M1 model migration, driven by M2 model changes. In this paper we present the classification of M2 model changes, in regard to their impact on corresponding M1 models. To illustrate our considerations, we use the example of super type associations in context of the Eclipse Modeling Framework.

1 Introduction

In the field of Model Driven Development, the modeling infrastructures are providing the substrate for modeling tools, required in order to build and maintain the modeling artifacts. The meta-meta-model (M3 model) is at the heart of a modeling infrastructure and is assumed to be fixed for a reasonable amount of time. In recent years, the Eclipse Modeling Framework[BSM+03] has established itself as de facto standard for modeling infrastructures. The Ecore is the M3 model corresponding to the Eclipse Modeling Framework. The Ecore M3 model is comparable to the EMOF standard from OMG[Obj04]. Because the Ecore model contains itself to the essential meta-modeling construct, it's well suited to demonstrate the ideas of M2 model changes classification.

The classification of detected M2 changes is in our view the most crucial step in the M1 model migration process. This step is preceded by the detection of M2 changes, which delivers a set of changes between two revisions of a M2 model. The classification step is succeeded by the user input gathering for changes, which can not be resolved automatically. Therefore, the classification step determines the amount of user effort, required to enable the M1 model migration. It is out goal, to minimize the number of user operations, required to migrate the M1 models.

In this paper, we use the EMF type hierarchy representation of EMF, to clarify out view on M2 model changes classification. Figure 1 depicts the EMF type hierarchy. All super types of a given class, are represented as links of the ordered eSuperTypes link set.

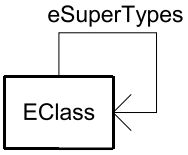


Figure 1: M3 EMF type hierarchy.

The rest of this paper is structured as follows. In section 2 we present the classification scheme for M2 model changes. In section 3 we explain the classification presented in section 2 on the example of eSuperType reference. Finally, we conclude and provide an overview of the future work in section 4.

2 Proposed Changes Classification Scheme

In order, to allow for the M1 model migration, the changes performed on the M2 model have to be partitioned into sets, which are relevant to the already existing M1 instances and those, which do not have an impact. Therefore, the most rudimentary classification is the partitioning of the changes set into this two partitions. We call the set of changes with impact on M1 instances, the breaking changes, while the latter set of changes is called additive.

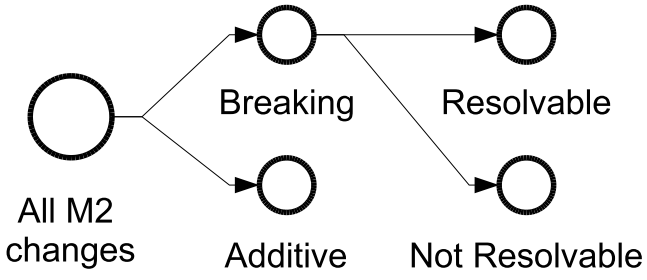


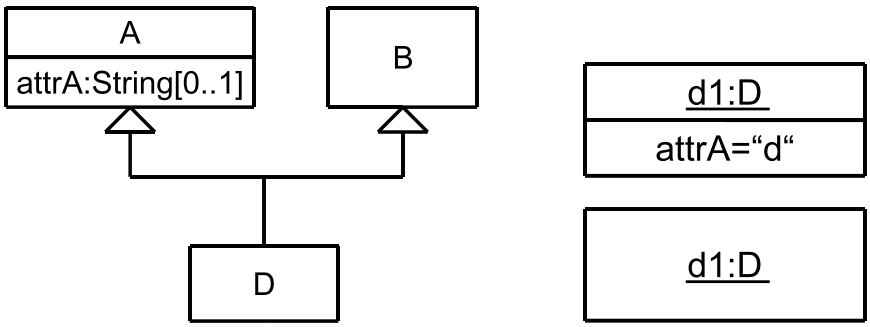
Figure 2: Proposed M2 changes classification.

The classification into the two presented sets, is useful in regard to additive changes. These changes can be simply ignored by the migration. The set of breaking changes requires action, but can not be worked on as is, because some of the changes require human assistance, to capture the semantic of the changes. We therefore propose, to further partition the breaking changes into the following two categories. The breaking and resolvable changes, are changes which can be migrated without further human assistance. The breaking and not

resolvable changes, are requiring further inputs, to migrate the concerned M1 instances. Figure 2 depicts the proposed M2 changes classification.

3 Example of Changes Classification

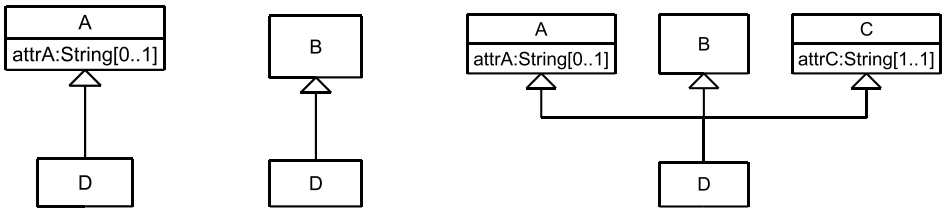
To improve the understanding of the provided M2 model changes classification, we present an example of all three change classes. The example is based upon the eSuperTypes relation, already presented in section 1.



(a) (b)

Figure 3: Version 1 of the example M2 model and corresponding instances.

The example M2 model is depited in figure 3(a). The example M2 model contains three classes A, B and D. The classes A and B are supertypes of class D. Further, the figure 3(b) depicts two instances of the example M2 model.



(a) (b) (c)

Figure 4: Three variants of M2 model changes.

The presented instance are valid, as long as the author does not change the M2 model. To

demonstrate the proposed changes classification, we will use the changes introduced to the M2 model, to obtain the models depicted in figure 4.

The change depicted in figure 4(a) constitutes the deletion of class B. This class did not have any attributes and therefore, no illegal attributes can exist in its instances. Therefore, this change is additive. The change introduced in figure 4(b) consists of class A deletion. Class A carried the attrA attribute which was optional. It is possible, that instances of class D, carrying values of attrA are existent and now became invalid. En example of such an instance is the upper insance shown in figure 3(b). This change therefore is breaking. However, the values of attrA can be deleted from instances of D, without human intervention. Therefore, this change is breaking and resolvable. The figure 4(c) depicts a change, where a new class C has been introduced. This class carries a mandatory attribute attrC. Because attrC is mandatory, no valid instances of D can possibly exist. This fact makes this change breaking and not resolvable, because human intervention is needed to provide values for attrC.

The provided cases are not exhaustive, however they do illustrate all three proposed changes categories. The classifications do not take into account known M1 instances. Therefore, an open world, where arbitrary, unknown M1 instances of the changed M2 model may exist, is being assumed. If this assumption is can be dropped and a closed world, where all existing M1 instances are known, can be assumed, some of the changes may be moved to less problematic categories. For example, if the only M1 instance in existence is the lower instance from diagram 3(b), the change from figure 4(b) becomes additive, because no instance can possibly be broken by it.

4 Conclusion

In this paper we have shown a possible approach to the classification of M2 model changes. This work is being performed in larger context of M1 model migration. The presented approach is being constructed, in order to allow for effort minimization during M1 model migration. Therefore, the presented step is crucial to the success of the overall approach. Now we work on the complete categorization of possible Ecore model changes, according to the presented scheme. We are convinced, that the categorization of M2 model changes, will make the semi automatic M1 model migration a feasible option for large scale software engineering projects.

References

[BSM+03] Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick und Timothy Grose. *Eclipse Modeling Framework*. Addison Wesley Professional, 2003.

[Obj04] Object Management Group. *Meta Object Facility (MOF) 2.0 Core Final Adopted Specification*, 2004.