(co-)evolution in MDSE ecosystems

Mengerink, J.G.M.

Published in:
Benevol 2014 (Seminar on Software Evolution in Belgium and the Netherlands, Amsterdam, The Netherlands, November 27-28, 2014)

Published: 01/01/2014

Document Version
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
(Co-)Evolution in MDSE ecosystems

J.G.M. Mengerink
Eindhoven University of Technology, The Netherlands
Email: j.g.m.mengerink@tue.nl

INTRODUCTION

In model driven software engineering (MDSE) [10], model-transformations are central artifacts [14]. They depend on meta-models for their structure and relate the different models in the ecosystem. However, meta-models evolve, for instance because of new insights in the systems they model. A pressing issue in industry, is that maintaining model-transformations with respect to meta-model evolution is very costly [3] in both a time-related and skill-related sense. To this end, it is desirable to automate this co-evolution of transformations, with respect to meta-model evolution, to the furthest extent possible. Although for meta-model/model co-evolution, a variety of tools exist [15], for meta-model/model-transformation co-evolution, most tools remain in prototype [2], [4], [12]. The methods and techniques of these prototypes are promising. However, the prototypes are all aimed towards specific use-cases and only offer support that is sufficient for their specific use-cases. When one requires to evolve artifacts that are not in-line with the artifacts in those case-studies, these prototypes are not yet mature enough.

In this extended abstract we sketch the envisioned direction of the PhD research addressing the (co-)evolution challenge in MDSE ecosystems. The research is to be conducted in 2014–2018.

Fig. 1: Abstract representation of evolution in an MDSE ecosystem, extended from the non-evolutionary variant in [8]

INDUSTRIAL CONTEXT

Our research takes place at ASML, the leading provider of complex lithography systems. Here we have access to an industrial repository containing a large MDSE ecosystem with version history going back up to three years. Our ecosystem can be represented similarly to that of Jouault and Kurtev [8]. However, we are more interested in the evolutionary axis through such a system, as is illustrated in Figure 1. As in the non-evolution version [8], our representation shows two models (α .MMA and β .MMB) relating to meta-model MMA and MMB respectively. To incorporate evolution, we include the evolved versions of MMA and MMB (MMA’ and MMB’ respectively), to which evolved models α’.MMA’ and β’.MMB’ conform. Lastly, our model-transformation A2B.qvto should co-evolve to support the new models, leading to A’2B’.qvto.

RESEARCH QUESTION

The main question that we aim to solve is how to specify the differences between difference versions of our modeling artifacts (meta-models, models, and model-transformation). That is: in what way can we specify, for example, δMMA, such that we have enough information to co-evolve the related models and model-transformations. This specification can take place either before, or after evolution of the primary artifacts (i.e. the meta-models). If one was to provide such a specification a-priori, it could be used to perform evolution on both the primary, and the secondary artifacts (i.e. the model-transformations). Alternatively, this specification could be created after evolution of the primary artifacts (potentially in an automated way), and used solely for the evolution of secondary artifacts.

RELATED WORK

In literature, a number of different approaches into specifying evolution have been addressed. State-based approaches attempt to calculate the difference between two versions of a meta-model (δMMA), then adapt the related artifacts (A2B.qvto and α .MMA). Often, these approaches attempt to aggregate smaller changes into higher order transformations (HOTs). [1], [5], [17]

Generation approaches aim to fully generate model-transformation, rather than evolving them from previous versions. By-example techniques can be employed, letting the user specify relations between model instances (i.e. between α’.MMA’ and β’.MMB’) [9]. Using this information, A’2B’.qvto is generated, rather than evolved from A2B.qvto. Other approaches include regenerating from a shared ontology of concepts [16].
**Operator-based** approaches define a set of operators which the developer can use. These operators affect both the metamodel and artifacts, while preserving conformance during the evolution. Rather than compute $\delta_{\text{MMA}}$, the user creates it by the successive applications of these operators. While an extensive set of these operators exists for model co-evolution [6], only a very restricted set is available for transformation co-evolution [11].

An example of an operator-based language is one by Luo [13]. However, it focuses on refinement, only allows for additive changes, and does not consider subtractive changes [7] (i.e. removal of elements). Furthermore, this approach specifies change at a fine-grained level of detail. To effectively co-evolve artifacts, it is desirable that changes are specified at a higher, more coarse, level. For example, specifying change in terms of adding and deleting model elements, provides little information about the intent of the user. However, if one were to specify change in terms of higher-order operations such as Extract Superclass or Flatten Hierarchy, additional information with respect to the evolution process can be obtained (i.e. to what end is the user adding/removing a certain element?). Using this additional information, artifacts can be co-evolved more precisely, such that the result is closer to the end-result desired by the user.

In order to extend such a language with subtractive and update (e.g. renaming an element) operations [7], the different operations (either low-level or high-level) need to be categorized with respect to the context in which they operate. For instance, extending a meta-model with an optional element, does not require conforming models to be update, so $\alpha_{\text{MMA}} = \alpha'_{\text{MMA}}$. We wonder whether we can discover, and use these properties to facilitate co-evolution.

**Envisioned Approach**

Given the large amount of available work for operator-based (co-)evolution of models [6], we feel research in to operator-based (co-)evolution of model-transformations will be the most fruitful. The first aim of our study will be to increase the available operators for model-transformations, by looking at the available operators of models. In this way, we aim to specify the difference between two meta-model versions in terms of these operators. An added benefit to this approach is that such a sequence of operators should immediately give us a specification for co-evolution of model-transformations. However, rather than creating these operators in just a, traditional, bottom-up fashion, additionally we will attempt use extract operators from the ASML repositories. Secondly, our research will focus on semi-automatic reconstruction of operator-sequences from a difference specification between meta-model versions. The latter should close the gap between state-based and operator-based approaches.

**References**


