Udapt

Edapt Extensions for Industrial Application

Josh G.M. Mengerink* Alexander Serebrenik* Mark van den Brand* Ramon R.H. Schiffelers*,†

*Eindhoven University of Technology, the Netherlands and †ASML, the Netherlands
{j.g.m.mengerink, aserebrenik, m.g.j.v.d.brand}@tue.nl, ramon.schifflers@asml.com

Abstract

Domain specific languages (DSLs) allow modeling systems in terms of domain concepts. We discuss the shortcomings of existing DSL evolution approaches when applied in industry and propose extensions addressing these shortcomings.

Categories and Subject Descriptors D.2.7 [*Software Engineering*]: Distribution, Maintenance, and Enhancement; I.6.5 [*Simulation and modeling*]: Model Development

Keywords Model Driven Engineering, Models, Evolution

1. Introduction

DSLs are often designed using meta-models, which evolve over time (Favre 2005) (*e.g.,* due to new demands from the domain). Evolution of the meta-models can trigger *co-evolution* of models conforming to these meta-models (Cicchetti et al. 2009). In industry, the models can number in the thousands (Mengerink et al. 2016), making manual model co-evolution infeasible.

Based on earlier comparative studies (Rose et al. 2009; Herrmannsdörfer and Wachsmuth 2014), we have performed an internal evaluation of various automated model migration tools at ASML (ASM). We found that existing tools are no longer available, or not mature enough. Of the two mature tools remaining, Flock (Rose et al. 2010) was experienced as having a high learning curve. Edapt, however, met all the requirements for the ASML case (Mengerink et al. 2016; Vissers et al. 2016), and has been previously applied in a number of industrial case studies (Herrmannsdörfer et al. 2009, 2008). We have thus selected Edapt as the primary candidate tool for meta-model/model co-evolution.

ITSLE'16, October 31, 2016, Amsterdam, Netherlands
© 2016 ACM. 978-1-4503-4646-7/16/10...$15.00
http://dx.doi.org/10.1145/2998407.2998409

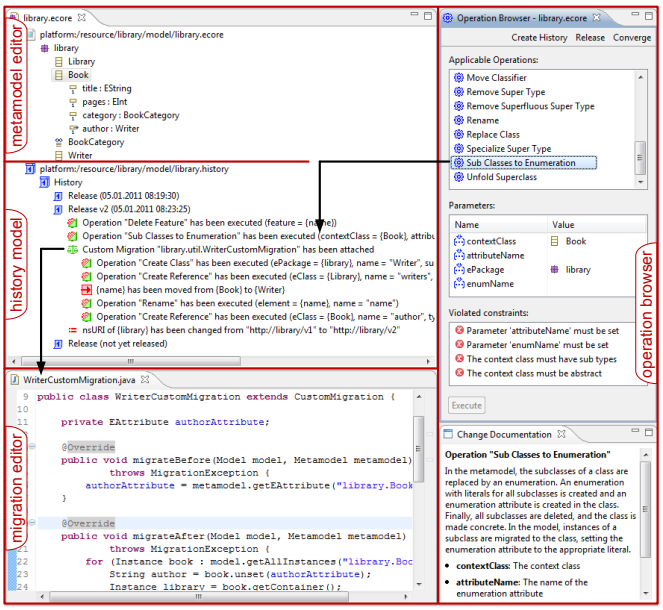

Figure 1. Edapt, taken from (Herrmannsdörfer 2011)

2. Edapt

Edapt, previously known as COPE (Herrmannsdörfer 2011), encodes patterns of meta-model evolution and model co-evolution into *operators*. Users can apply the operators to specify evolution and co-evolution. The specification is then embedded in a modeling workbench by means of a *migrator*.

Experience at ASML has revealed several deficiencies in the usability of Edapt. We require Edapt to allow as much reuse of (co-)evolution knowledge as possible (**Req.1**), not to disrupt the developer's workflow (**Req.2**), deal with the large meta-models and large amounts of models (**Req.3**) and provide fast feedback on the effect of a (co-)evolution specification on the models (Muşlu et al. 2015) (**Req.4**).

Firstly, Edapt supports the creation of custom operators, when no reusable operator is available. However, these custom operators can only be applied once, and are not transferable to other users (**Req.1**). The only way of adding reusable

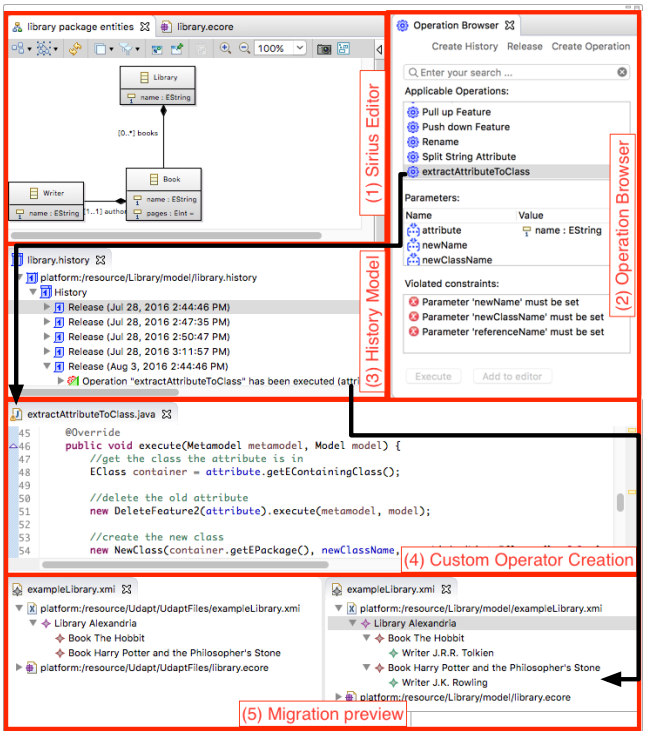

Figure 2. Udapt.

operators to Edapt, is by modifying its source code and recompiling it, which interrupts the development flow (**Req.2**).

Secondly, the graphical representation of the meta-model being evolved does not scale with the meta-model as the default tree editor (the meta-model editor in Figure 1) becomes hard to use for large meta-models (**Req.3**).

Thirdly, when co-evolving models every model has to be individually opened in Edapt. In an industrial context, where models number in the thousands, this is infeasible (**Req.3**).

Lastly, when specifying meta-model evolution, it is unclear what the effect operators have on the models due to lack of feedback on the model level (**Req.4**).

3. Udapt

Udapt is an extension of Edapt, improving its usability.

Firstly, we have simplified the process of creating reusable operators. In Udapt, custom operators can be defined in the same way as in Edapt. However, in Udapt, custom operator are reflectively compiled (Figure 2: 2,4) and can be used in the specification of the migrator. These operators are stored as file, and can be shared among developers (**Req.1, Req.2**).

Secondly, to solve the scalability issues of the meta-model editor, we have integrated a Sirius (Viyović et al. 2014) meta-model editor into Edapt (Figure 2:1). This allows the user to edit their meta-models in a more intuitive way, which scales better with large meta-models (**Req. 3**).

Thirdly, we have implemented a mass-migrator that allows users to migrate multiple models, reducing overhead for developing intermediate versions of DSLs (**Req. 3**).

Lastly, users are now able to add sample models as "preview" (Figure 2:5). When the migrator is modified, these "preview" models are automatically migrated. Subsequently, both the original and co-evolved models are rendered to give insight into the effect the migrator has on models (**Req. 4**).

4. Future Work & Conclusion

As future work, we would like to conduct industrial user studies to evaluate Udapt.

Acknowledgments

The authors thank Theo Baart, Rick Bouten, Dennis Brons, Stefan Habets, Thijs Ledeboer, Wout de Ruiter, Bart van der Vecht, Bart Verberne, and Leroy Visser, who implemented Udapt as part of their undergraduate capstone project.

References

ASML. `http://www.asml.com/`. Accessed: 2015-04-07.

A. Cicchetti, D. Di Ruscio, and A. Pierantonio. Managing dependent changes in coupled evolution. In *ICMT*, volume 5563 of *LNCS*, pages 35–51. Springer, 2009.

J.-M. Favre. Languages evolve too! changing the software time scale. In *Principles of Software Evolution*, pages 33–42, 2005.

M. Herrmannsdörfer. COPE - A workbench for the coupled evolution of metamodels and models. In *SLE*, volume 6563 of *LNCS*, pages 286–295. Springer, 2011.

M. Herrmannsdörfer and G. Wachsmuth. Coupled evolution of software metamodels and models. In *Evolving Software Systems*, pages 33–63. Springer, 2014.

M. Herrmannsdörfer, S. Benz, and E. Juergens. Automatability of coupled evolution of metamodels and models in practice. In *MoDELS*, pages 645–659. Springer, 2008.

M. Herrmannsdörfer, D. Ratiu, and G. Wachsmuth. Language evolution in practice: The history of GMF. volume 5969 of *LNCS*, pages 3–22. Springer, 2009.

J. G. M. Mengerink, A. Serebrenik, R. R. H. Schiffelers, and M. G. J. van den Brand. A complete operator library for DSL evolution specification. In *ICSME*, 2016.

K. Muşlu, Y. Brun, M. D. Ernst, and D. Notkin. Reducing feedback delay of software development tools via continuous analysis. *TSE*, 41(8):745–763, 2015.

L. M. Rose, R. F. Paige, D. S. Kolovos, and F. A. C. Polack. An analysis of approaches to model migration. In *MoDSE-MCCM Workshop*, pages 6–15, 2009.

L. M. Rose, D. S. Kolovos, R. F. Paige, and F. A. Polack. Model migration with Epsilon Flock. In *ICMT*, volume 6142 of *LNCS*, pages 184–198. Springer, 2010.

Y. Vissers, J. G. M. Mengerink, R. R. H. Schiffelers, A. Serebrenik, and M. Reniers. Maintenance of specification models in industry using Edapt. In *FDL*, 2016.

V. Viyović, M. Maksimović, and B. Perisić. Sirius: A rapid development of dsm graphical editor. In *International Conference on Intelligent Engineering Systems*, pages 233–238, 2014.