Applying Domain-Specific Languages in Evolving Product Lines

Juha-Pekka Tolvanen
MetaCase
Jyväskylä, Finland
jpt@metacase.com

Steven Kelly
MetaCase
Jyväskylä, Finland
stevek@metacase.com

ABSTRACT

This demonstration shows how domain-specific languages for modeling and generating variant products can evolve together with the product line. In the demonstration, examples from practice are illustrated and executed, covering both domain engineering and application engineering. The examples cover the typical evolution scenarios: adding new features and variability points to a product line and then to existing products, changing their variation, and removing them completely from the product line. The evolution of the domain-specific languages, and the versioning of both the languages and products built with the languages, are demonstrated.

CCS CONCEPTS

• **Software and its engineering → Software product lines**; **Domain specific languages**; *Software configuration management and version control systems*; Model-driven software engineering.

KEYWORDS

Software Product Line, product derivation and generation, Domain-Specific Language, Domain-Specific Modeling, MetaEdit+

ACM Reference Format:
Juha-Pekka Tolvanen and Steven Kelly. 2019. Applying Domain-Specific Languages in Evolving Product Lines. In *23rd International Systems and Software Product Line Conference - Volume B (SPLC '19), September 9–13, 2019, Paris, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3307630.3342389

1 INTRODUCTION

Product lines are constantly evolving: new features and their variability points will emerge and existing ones change or even become obsolete. At the same time, products that have already been delivered need to be maintained and updated with new features — including features not known when these products were originally delivered. Domain-Specific Languages (DSLs) can be used to cover the rich variation space within product lines [1]. With a DSL, the language definition itself defines the variation space and related rules. Language users then automatically follow the variation space when creating variant products. Domain engineers define the language and application engineers use it. In a fair number of cases the final products can be automatically generated from the high-level variant specifications [1, 5].

DSLs, and in particular related tools, have not always recognized the need for evolution nor provided tool support for it. In such cases, once a DSL has changed there has been a huge and often manual effort to update all the existing specifications to the new language version[1]. This unfortunate situation is the norm with textual DSLs and their syntax-oriented editors, but also found in many graphical modeling tools, including recently created ones (e.g. Eclipse-based [4]). In this demonstration we will show how a modern, mature tool, the MetaEdit+ language workbench, recognizes the importance of evolution and provides tool support for managing it. This demonstration focuses on features of MetaEdit+ 5.5 SR1.

2 METAEDIT+ FOR EVOLVING PRODUCT LINES

With MetaEdit+ Workbench [3], domain engineers define DSLs and get modeling tools for a specific product line in a fraction of the time needed by other tools [2]. First, experts define a domain-specific language as a metamodel containing the domain concepts and rules, and then specify the mapping from that to code by defining generators. DSLs can be collaboratively developed and maintained as well as versioned — using the functionality available in MetaEdit+. Language creation in an industrial setting takes on average 10 working days [7].

MetaEdit+ Modeler follows the defined modeling language and automatically provides full modeling tool functionality: editors, browsers, generators, collaborative model editing etc. Application engineers create variants by editing designs and by executing generators producing the product variants: their source code, configuration, deployment, tests, documentation etc. This leads to 5–10 times faster product derivation and time-to-market compared to traditional manual practices [6].

The use of DSLs, however, does not end once the first language version is created. Instead, DSLs need to be refined, enhanced and maintained along with the other assets of the product line. In particular, not only does the DSL (language and generators) evolve, but also changes to the language need to be reflected — ideally automatically and unobtrusively to language users — to the models specifying product variants. MetaEdit+ provides functionality to manage such evolution with its automatically collected history of models and all changes made to them. Both evolving DSLs and the variant specifications that have been created can be versioned to existing version control systems (e.g. Git or SVN). In the demonstration these tool features are shown in detail, following evolution scenarios from practice.

[1] One public example of this is the evolution of AUTOSAR metamodel.

SPLC '19, September 9–13, 2019, Paris, France

ACM ISBN 978-1-4503-6668-7/19/09…$15.00
https://doi.org/10.1145/3307630.3342389

3 CASES OF PRODUCT LINE EVOLUTION

In the demonstration we show the language definition and language usage scenario within different product lines, including Internet of Things applications, industrial automation systems and consumer products. The evolution of the language is addressed based on two dimensions:

(1) Nature of change: adding, changing or removing parts from the product line and thus from the DSL
(2) Whether the change influences only to the language definition, to the generator used for product derivation or to both.

Each case starts with presenting the results of domain engineering: DSL and generators. This is followed by showing DSL use in application engineering and generating product variants. Next, we present a scenario of product line evolution that calls for changes in the DSL. These changes are then discussed and implemented during the demonstration: changing the language elements, constraints, concrete syntax or generators.

Based on these language changes, a new version of the DSL is delivered to the language users, i.e. application engineers. With MetaEdit+ they can view the model history, inspect the changes made and version the product variants to version control systems like GitHub, Bitbucket etc. The same versioning approach is also available for domain engineers (for DSLs and related generators). The demonstration ends by showing the generated application code, its execution, or other artifacts relevant for the product variants developed.

REFERENCES

[1] Czarnecki, K., Eisenecker, U., Generative Programming, Methods, Tools, and Applications, Addison-Wesley, 2000.
[2] El Kouhen, A., Dumoulin, C., Gerard, S., Boulet, P., Evaluation of Modeling Tools Adaptation, 2012, https://hal.archives-ouvertes.fr/hal-00706701v2
[3] MetaCase, MetaEdit+ 5.5 User's Guides, 2017, https://www.metacase.com/support/55/manuals/
[4] Rocco Di, J., Ruscio Di, D., Narayanankutty, H., Pierantonio, A., Resilience in Sirius Editors: Understanding the Impact of Meta-Model Changes, Models and Evolution Workshop at Models Conference, 2018 (http://www.models-and-evolution.com/2018/papers/8.pdf)
[5] Tolvanen, J.-P., Kelly, S., Defining Domain-Specific Modeling Languages to Automate Product Derivation: Collected Experiences. Proceedings of the 9th International Software Product Line Conference, Springer-Verlag, 2005.
[6] Tolvanen, J-P. and Kelly, S. Model-Driven Development Challenges and Solutions - Experiences with Domain-Specific Modelling in Industry. In Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development, SCITEPRESS Science and Technology Publications, Lda, 2016
[7] Tolvanen, J.-P., Kelly, S., Effort Used to Create Domain-Specific Modeling Languages. In ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems (MODELS 18), ACM, New York, NY, USA, 2018