MetaEdit+: Defining and Using Domain-Specific Modeling Languages and Code Generators

Juha-Pekka Tolvanen
MetaCase Consulting
Ylistonmaentie 31
FIN-40500 Jyvaskyla, Finland
+358 14 4451406
jpt@metacase.com

Matti Rossi
Helsinki School of Economics
FIN-00100 Helsinki, Finland
+358 9 43138996
mrossi@hkkk.fi

ABSTRACT
MetaEdit+ is an environment that allows building modeling tools and generators fitting to application domains, without having to write a single line of code. The capability to define modeling tools and generators is relevant as it provides the ability to raise the abstraction of design work from code to domain concepts, and a raise in abstraction leads to an imminent raise in productivity, as illustrated by the past years' experiences.

In domain-specific modeling and MetaEdit+, one expert defines a domain-specific language as a metamodel containing the domain concepts and rules, and specifies the mapping from that to code in a domain-specific code generator. For the method implementation, MetaEdit+ provides a metamodeling language and tool suite for defining the method concepts, their properties, associated rules, symbols, checking reports, and generators.

Once the expert defines a modeling method, or even a partial prototype, the rest of the team can start to use it in MetaEdit+ to make models with the modeling language and the required code is automatically generated from those models. Based on the metamodel, MetaEdit+ automatically provides CASE tool functionality: diagramming editors, browsers, generators, multi-user/project/platform support, etc.

The MetaEdit+ demo will focus on showing how the domain-specific languages and generators are made; complete with several examples of domain-specific methods and related code generators.

Categories and Subject Descriptors
D 2.2 [**Design Tools and Techniques**]: *Computer-aided software engineering* (CASE)

General Terms: Design, Languages.

Keywords
Metamodel; domain-specific modeling; code generators**.**

1. INTRODUCTION
'One tool fits all' – that seems to be the common principle followed by many tools on the market today. Alternatively, MetaEdit+ is an environment that enables you to build your own modelling tools and code generators fitting to your own domain — without having to write a single line of code.

OOPSLA'03, October 26–30, 2003, Anaheim, California, USA.
ACM 1-58113-751-6/03/0010.

The capability to define modelling and generator tools is relevant as it provides the possibility to raise the abstraction of design work from code to domain concepts, and a raise in abstraction leads to an imminent raise in productivity, as illustrated by the past years' experiences, such as those listed in the SEI Product Line Hall of Fame. Empirical studies such as [1] consistently back up this observation.

With these benefits, it is little wonder that there is a growing interest in domain-specific modelling, as shown by workshops at OOPSLA [2] and this year's theme of Domain-Driven Development.

2. DOMAIN-SPECIFIC MODELLING
In domain-specific modelling and MetaEdit+, one expert defines a domain-specific language containing the domain concepts and rules, and specifies the mapping from that to code in a domain-specific code generator. As soon as the expert defines a modelling method, or even a partial prototype, the team can start to use it in MetaEdit+ to make models with the modelling language and code is automatically generated from those models. Developers no longer need to solve the problem of manually mapping domain ideas into quality code by themselves, time after time. As the modelling language is based on the already known and used domain concepts and rules, it is easy to remember and understand by all developers.

3. METACASE TECHNOLOGY
For method implementation, MetaEdit+ provides a metamodeling language and tool suite for defining the method concepts, their properties, associated rules, symbols, checking reports and generators with ease [3]. The method definition is stored as a metamodel to the MetaEdit+ repository allowing future modifications, which reflect automatically to models and generators.

MetaEdit+ follows the given method definition and automatically provides full CASE tool functionality: diagramming editors, browsers, generators, multi-user/project/platform support, etc. Whole team can immediately start to edit designs as graphical diagrams, matrices or tables, switching between views according to user needs. User can browse designs with filters, apply components, link models to other designs following domain rules, and check models with various pre/user-defined reports. The results of modelling can be published to the web or word processors, and generated into code for your product.

4. CODE GENERATION

In contrast to the generic code generators provided with standard CASE tools, the basis of code generation in domain-specific modeling is the domain itself. As with product line engineering [4] the architecture and patterns of code found in implementations for that domain are analyzed to determine code commonalities and variabilities over a product family [5].

The variabilities form a significant source of information when designing what information needs to be stored in models. For each variability point, there must be a corresponding point in a model where information can be stored about the choice of value for this product variant. The code generator's task is to transform the models into code, often largely in the form of calls to components using these values as arguments.

Commonalities are abstracted out into framework code: a layer of code between the generated code and the platform and standard libraries [6]. This information is thus not included as part of the models — why should every model include something that is the same for all models? Instead, the framework code is linked in with that generated from the models.

5. CONCLUSION

Domain-specific modeling provides significant increases in productivity, especially for product families. Providing tool support for such a modeling method has previously required at least a man-year of work. A metaCASE tool such as MetaEdit+ reduces the time needed down to the order of days or weeks. Industrial experiences such as Nokia [7] show productivity gains of 5-10 times, and comparable decreases in the time needed for new users to become productive.

6. REFERENCES

[1] Kieburtz, R. et al., A Software Engineering Experiment in Software Component Generation, Proceedings of 18th International Conference on Software Engineering, Berlin, IEEE Computer Society Press, March, 1996.

[2] OOPSLA Workshop on Domain-Specific Visual Languages (DSVL'01), Juha-Pekka Tolvanen, Steven Kelly, Jeff Gray, Kalle Lyytinen (eds.), University of Jyväskylä 2001.

[3] Kelly, S., Lyytinen, K., Rossi, M., MetaEdit+: A fully configurable multi-user and multi-tool CASE and CAME environment, Advanced Information Systems Engineering, proceedings of the 8th International Conference CAISE'96, Constantopoulos et al (Ed.), Springer-Verlag, 1996.

[4] Weiss, D., Lai, C. T. R., Software Product-line Engineering, Addison Wesley Longman, 1999.

[5] Domain-Specific Application Frameworks, Mohamed E. Fayad and Ralph E. Johnson (Eds.), Wiley 1999.

[6] Pohjonen, R., and Kelly, S., "Domain-Specific Modeling," Dr. Dobbs Journal, August 2002.

[7] MetaCase, Benefits of MetaCASE: Nokia Mobile Phones Case Study, http://www.metacase.com/papers/

ACKNOWLEDGEMENTS

The authors would like to thank Slava Arion and Steven Kelly for their help in the preparation of this article.