

# **MetaEdit+: Domain-Specific Modeling and Code Generation Environment**

Slava Arion, Juha-Pekka Tolvanen  
MetaCase, Ylistonmaentie 31  
FIN-40500 Jyväskylä, Finland  
{slava, jpt}@metacase.com  
<http://www.metacase.com>

**Abstract.** The MetaEdit+ demonstration will focus on showing how the domain-specific languages and generators are defined; complete with examples from different real-world cases.

## **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 modeling tools and code generators fitting to your own domain — without having to write a single line of code.

The capability to define modeling 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 [1], [6].

## **2 Domain-Specific Modeling**

In domain-specific modeling 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 [5]. As soon as the expert defines a modeling method, or even a partial prototype, the team can start to use it in MetaEdit+ to make models with the modeling 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 modeling 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. 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 modeling 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 the architecture and patterns of code found in implementations for that domain are analyzed to determine code commonalities and variabilities over a product family [4], [7].

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 [5], [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 [2], [3] show productivity gains of 5-10 times, and comparable decreases in the time needed for new users to become productive.

## References

1. Kieburtz, R. et al., A Software Engineering Experiment in Software Component Generation. In: Proceedings of 18th International Conference on Software Engineering, Berlin, IEEE Computer Society Press, March (1996)
2. MetaCase, Nokia Mobile Phones Case Study, <http://www.metacase.com/papers/> (1999)
3. MetaCase, Automated product family development: Nokia Tetra terminals, <http://www.metacase.com/papers/> (2003)
4. Fayad, M., Johnson, R., (Eds.), Domain-Specific Application Frameworks, Wiley (1999)
5. Pohjonen, R., and Kelly, S., "Domain-Specific Modeling," Dr. Dobbs Journal, August (2002)
6. Tolvanen, J.-P., Keeping it in the family, Application Development Advisor, July-August, 101 Communications (2002)
7. Weiss, D., Lai, C. T. R., Software Product-line Engineering, Addison Wesley Longman (1999)