MetaEdit+:
Integrated modeling and metamodeling environment for domain-specific languages

Juha-Pekka Tolvanen
MetaCase
Ylistonmaentie 31
FI-40500 Jyvaskyla, Finland
+358 14 4451400
jpt@metacase.com

Abstract
Domain-Specific Modeling (DSM) raises the level of abstraction beyond programming by specifying the solution directly using domain concepts. In many cases, the final products can be generated from these high-level specifications. This automation is possible because both the language and generators need fit the requirements of only one company and domain.

This demonstration illustrates DSM by showing real world cases from various fields of software development. These cases describe how DSM, giving first class support for modeling, can prevent incorrect or unwanted designs at the early stages of development, and how full code can be generated from the modeler's point of view. Second part of the demonstration will show in an interactive manner both the design side and the use side of DSM languages and generators. Using MetaEdit+ tool for metamodeling, we define a DSM for a given domain and apply it to generate full code from high-level models.

Categories and Subject Descriptors D 2.2 [**Design Tools and Techniques**]: Computer-aided software engineering (CASE) D 2.6 [**Programming Environments**]: Graphical environments D 3.2 [**Language Classifications**]: Design languages, specialized application languages, very high-level languages

General Terms Design, Languages

Keywords metamodel; domain-specific modeling; code generators

1. Introduction
Domain-Specific Modeling raises the level of abstraction and hides today's programming languages, in the same way that today's programming languages hide assembler [5, 6]. Symbols and language constructs in a domain-specific model map to things in the domain - the world in which the application is to run. Rather than having concepts and symbols that map one-to-one with the constructs of a programming language, each symbol can be worth of several lines of code. This offers a whole level of abstraction higher than with current modeling languages, such as UML. The properties that characterize the symbol can further elaborate different mappings to code, or the connections the symbol has to other symbols offer further mappings etc. The developer can therefore solve the problem only once by visually modeling the solution using only familiar domain concepts. The final products can be automatically generated from these high-level specifications with domain-specific code generators, aided where necessary by existing component code [2, 3, 4].

As the name suggests, Domain-Specific Modeling is only possible because of narrowing down the design space, often to a single range of products for a single company [1, 2]. 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. An experienced developer can state exactly what code is wanted from models in a given domain. Normal developers then make models with the modeling language and code is automatically generated. As an expert has specified the code generators, they produce products faster and with better quality than could be done by normal developers by hand [3]. The generated result will be free of most kinds of careless mistakes, syntax and logic errors.

Generally speaking, defining a language and generator is considered a difficult task: this is certainly true once building a language for everyone. The task eases considerably if you make it only for one problem domain in one company. This task becomes even easier if you can use a tool that that support both DSM development and use.

2. MetaEdit+ for DSM
MetaEdit+ is an integrated environment that allows building modeling tools and generators fitting to specific application domains, without having to write a single line of code. In 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 this DSM implementation, MetaEdit+ provides a metamodeling language and tool suite for defining the method concepts, their properties, associated rules, symbols, checking reports, and generators. The method definition is stored as a metamodel in the MetaEdit+ repository allowing future modifications, which reflect automatically to models and generators.

MetaEdit+ follows the given language definition and automatically provides full modeling tool functionality: diagramming editors, browsers, generators, multi-user/project/platform support, etc. A whole team can immediately start to edit designs as graphical diagrams, matrices or tables, switching between views according to user needs.

OOPSLA'06 October 22–26, 2006, Portland, Oregon, USA.
ACM 1-59593-491-X/06/0010.

3. Examples of DSM

Every domain is different, and so every DSM example is different. This demonstration shows real world cases of DSM from various fields of software development: enterprise application development into mobile phones, financial product definition into B2B J2EE web site and voice menu development into 8-bit microcontroller. These samples cover a wide range of code generation target languages, scripting languages, object-oriented languages and assembler. These cases illustrate how DSM, giving first class support for modeling, can prevent incorrect or unwanted designs at early stages of the development, how underlying platform complexity is hidden, and how full code can be generated from the modeler's point of view.

4. Creating DSM

Second part of the demonstration will show in an interactive manner both the design side and the use side of DSM languages and generators. On the DSM use side, we implement the OOPSLA conference registration application into a mobile phone. This is done by modeling in MetaEdit+ tool (Figure 1).

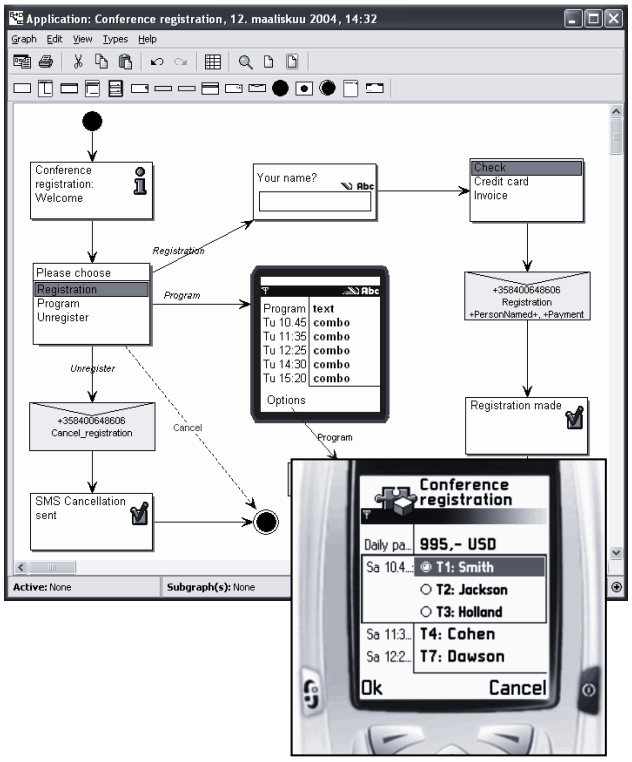

Figure 1. Sample model and generated application running

The design model is directly based on domain concepts, such as Note, Pop-up, SMS, Form, and Query. These are specific to mobile phone services and its user-interface widgets. As can be seen from the design model, all the implementation concepts are hidden. Developers can focus on finding the solution using the domain concepts. As the descriptions capture all the required static and behavioral aspects of the application, it is possible to generate the application fully from the models. In this case the generated code uses the services provided by the smartphone framework. After design, there is no need to map the solution to implementation concepts in code or in UML models visualizing the code. Nor there is need to change the generated code.

In the demonstration we shift next to the DSM creation side: Using MetaEdit+ tool for metamodeling, we extend the modeling language as well as the generator. Language extensions deal with adding domain constraints (Figure 2), rules and new concepts. Once the DSM is extended, this allows us to revert to modeling in order to finalize our sample conference registration application.

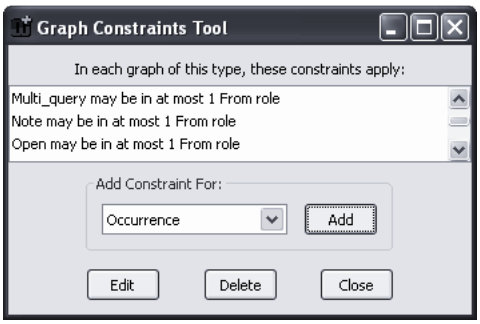

Figure 2. Adding constraint.

In addition to language and editor creation, the demonstration will show advanced features of DSM use. These include automatic update of models when the domain-specific language changes, a debugger for code generators integrated to the metamodel, generated code with live links back to the models, and the open architecture for integrating tools via XML or SOAP API to the metamodel and 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 [3, 4] show productivity gains of 3-10 times, and comparable decreases in the time needed for new users to become productive.

6. REFERENCES

[1] Fayad, M.E., Johnson, R. (Eds.), Domain-Specific Application Frameworks, Wiley 1999.

[2] Greenfield, J., Short, K., Cook, S., Kent, S., Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Wiley, 2004.

[3] 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.

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

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

[6] Tolvanen, J-P., Sprinkle, J., Rossi, M., (eds.), Proceedings of 5th OOPSLA workshop on Domain-Specific Modeling (DSM'05), University of Jyväskylä 2005.