The 6th OOPSLA Workshop on Domain-Specific Modeling

Juha-Pekka Tolvanen
MetaCase
Ylistonmentie 31
FIN-40500 Jyvaskyla, Finland
jpt@metacase.com

Jonathan Sprinkle
University of California, Berkeley
337 Cory Hall
Berkeley, CA 94720, USA
sprinkle@EECS.Berkeley.edu

Jeff Gray
University of Alabama at Birmingham
1300 University Blvd.
Birmingham, AL 35294 USA
gray@cis.uab.edu

Abstract
Domain-Specific Modeling raises the level of abstraction beyond programming by specifying the solution directly using visual models to express domain concepts. In many cases, final products can be generated automatically from these high-level specifications. This automation is possible because both the language and generators fit the requirements of only one domain. This paper introduces Domain-Specific Modeling and describes the related workshop.

Categories and Subject Descriptors D.3.2 [Languages]: Specialized application languages, very high-level languages; D.2.2 [Design Tools and Techniques]: Computer-aided software engineering (CASE)

General Terms Design, Languages

Keywords Modeling Languages; Metamodeling; Domain-Specific Languages; Code Generation

1. Introduction
An upward shift in abstraction often leads to a corresponding increase in productivity. In the past this has occurred when programming languages evolved towards a higher level of abstraction. Today, Domain-Specific Modeling (DSM) languages provide a viable solution for continuing to raise the level of abstraction beyond coding, making development faster and easier.

Industrial experiences of DSM consistently show it to be 5-10 times faster than current practices, including current UML-based implementations of MDA. As Bosch et al. [1] state, “the full value of MDA is only achieved when the modeling concepts map directly to domain concepts rather than computer technology concepts.” Accordingly, in DSM the models are constructed using concepts that represent things in the application domain, not concepts of a given programming language [9]. The modeling language follows the domain abstractions and semantics, allowing developers to perceive themselves as working directly with domain concepts. The models represent simultaneously the design, implementation and documentation of the system. In a number of cases, the final products can be generated automatically from these high-level specifications with domain-specific code generators. This automation is possible because of domain-specificity: both the modeling language and code generators correspond to the requirements of a narrow domain, often in a single company.

This paper introduces DSM by describing a general framework for defining domain-specific modeling languages and code generators for a specific purpose. This is followed by describing the focus and topics of the 6th OOPSLA workshop on Domain-Specific Modeling [12].

2. Defining and using domain-specific languages
Three things are necessary to achieve full automatic code generation from domain modeling: firstly, a modeling tool supporting a domain-specific modeling language; secondly, a code generator; and lastly, a domain-specific framework. Figure 1 shows these three elements at two levels: the definition level and the use level.

```
Language (Metamodel)  Code Generation  Framework
```

```
Models  Generate code  Domain Framework
```

Figure 1. Framework for domain-specific modeling

The top-level (representing the definition), is made once by the organization for a given domain. This forms the start-up cost of the DSM approach. Normally, one or two experts will define the modeling language (i.e., a metamodel) and related code generation, normally with a metamodelling tool [3, 6]. The metamodel is the implementation of the domain-specific modeling language, and includes the concepts and rules directly from the domain. The framework code will often have been made by developers in earlier projects in the domain, with some being added or modified specifically for the DSM creation project.

The bottom level process represents the use of a domain-specific modeling language and code generator. This level is performed many times, once for each product, by normal developers. Development time can often be further reduced by reusing parts of the DSM model which are common to several products. The code generation and use of domain framework or platform services require no effort by the developer. Together, these savings form the primary payback of the DSM approach.

This is unlike current visual modeling languages that are fixed to a specific notation that maps to semantically well-defined concepts of programming languages (like UML, SA/SD). Here, developers have to leap straight from requirements into
implementation concepts, and map back and forth between domain concepts, UML concepts, and program code. This requires significant time and resources, and can lead to errors.

In DSM, the specification models are built up of instances of the domain concepts from the language metamodel, in accordance with the rules. The code generation walks through the model and transforms the concept structures into code. In some cases the code will be fully self-contained; more often significant parts of the code will be calls to reusable components and domain framework. Since the code is generated, syntax and logic errors do not normally occur, and the resultant improvement in quality forms a significant secondary payback of the DSM approach [2].

3. Workshop focus and topics

DSM has been successfully applied in several different domains, including automotive manufacturing [7], digital signal processing [10], mobile devices [2], telecommunication [4, 11], and electrical utilities [4]. It is even claimed to replace currently used code-oriented methods [8]. However, more investigation is still needed in order to advance the acceptance and viability of domain-specific modeling.

The goals of the workshop are to collect and exchange experiences related to building and using DSMs; continue building and extending the DSM community; and address in focus groups the issues raised in the presented papers and at previous workshops. With 25 paper submissions, the 6th workshop aims to address topics both at practical and theoretical levels. On the practical side, the workshop includes topics dealing with application of modeling techniques within a specific domain. In addition to industrial projects, also descriptions of research ideas that initiate and forward the technical underpinnings of DSM are addressed. In particular, the importance of metamodelling is highlighted in this workshop. Metamodelling significantly eases the implementation of domain-specific languages and provides support for experimenting with the modeling language as it is built (thus, metamodel-based language definition also assists in the task of constructing generators that reduce the burden of tool creation and maintenance).

Some suggested topics of the workshop include:

- Tools for supporting domain-specific modeling (DSM)
- Metamodeling frameworks and languages
- Comparison and analysis of model-driven development approaches
- Principles for identifying constructs for DSM languages
- Industry/academic experience reports describing success or failure in using domain-specific modeling
- Novel approaches for code generation from domain-specific models
- Approaches to implement DSM languages
- Issues of support/maintenance of models and evolution of DSM language in accordance with the representative domain
- Version control techniques for DSMs
- Specific domains where this technology can be most productive in the future (e.g., embedded systems, product family domains or systems with multiple implementation platforms)
- Techniques for supporting model interchange between tools

4. REFERENCES