Levels of Independence in Aspect-Oriented Modeling

Jeff Gray, Yuehua Lin, Jing Zhang

University of Alabama at Birmingham
Department of Computer and Information Sciences
Birmingham, Alabama USA
{gray, liny, zhangj}@cis.uab.edu
http://www.gray-area.org

Abstract

We are investigating the application of aspect-oriented principles as an aid toward improving the capabilities of domain-specific modeling. The approach, in general, provides transformations by weaving modeling aspects into a base model. The modeling aspects provide variation points within the model and can be used to drive the synthesis of the model into different artifacts. In this brief position paper, we describe how our current and future work improves the capabilities offered in a model-based tool. We highlight several levels of independence that make the approach applicable to numerous modeling situations.

1. Introduction

... program structure should be such as to anticipate its adaptations and modifications. Our program should not only reflect (by structure) our understanding of it, but it should also be clear from its structure what sort of adaptations can be catered for smoothly. E. Dijkstra [2]

A longstanding goal in software development is to construct systems that are easily modified and extended. The desired result is to achieve modularization such that a change in a design/modeling decision is isolated to one location of a system. The proliferation of software in everyday life (e.g., embedded systems found in automobiles, mobile phones, and television sets) has increased the conformity and invisibility of software. The key characteristics of such software are its tight integration of information processing and the physical environment. As demands for such software increase, future requirements will necessitate new strategies for improved modularization in order to support the requisite adaptations.

Model-Driven Architecture (MDA) [4] is one such technology that many believe will be a leading force in improving adaptability in the software construction. Several of the concepts in the MDA are not new (e.g., the concept of platform independence has been around for a long time, and was particularly popularized in [15]), but the timing of several factors currently make it a viable technology. In the MDA, it is possible to model application functionality and non-functional aspects at higher levels of abstraction. It is also possible to model the interfaces among various components in terms of standard middleware. The result is middleware that is more flexible and robust. As Gerald Sussman observes, in traditional system development, "Small changes in requirements entail large changes in the structure and configuration" [13]. This statement is also true regarding model-based approaches. Often, a single change to a modeling element results in a super-linear production of generated code, for example. This is a big challenge facing modelers. Our research focuses on improving the changeability of domain modeling that contributes to rapid construction and evolution of models.

In the remaining part of this position paper, we give a brief introduction to the background of our research and then enter into a discussion of our current and future research goals.

2. Model Integrated Computing

Model Integrated Computing (MIC) is a model-based approach to software development, facilitating the synthesis of application programs from models created using customized, domain-specific program synthesis environments [9]. MIC employs domain-specific models to represent the software, its environment, and their relationship and thus, is well suited for the rapid design of complex computer based

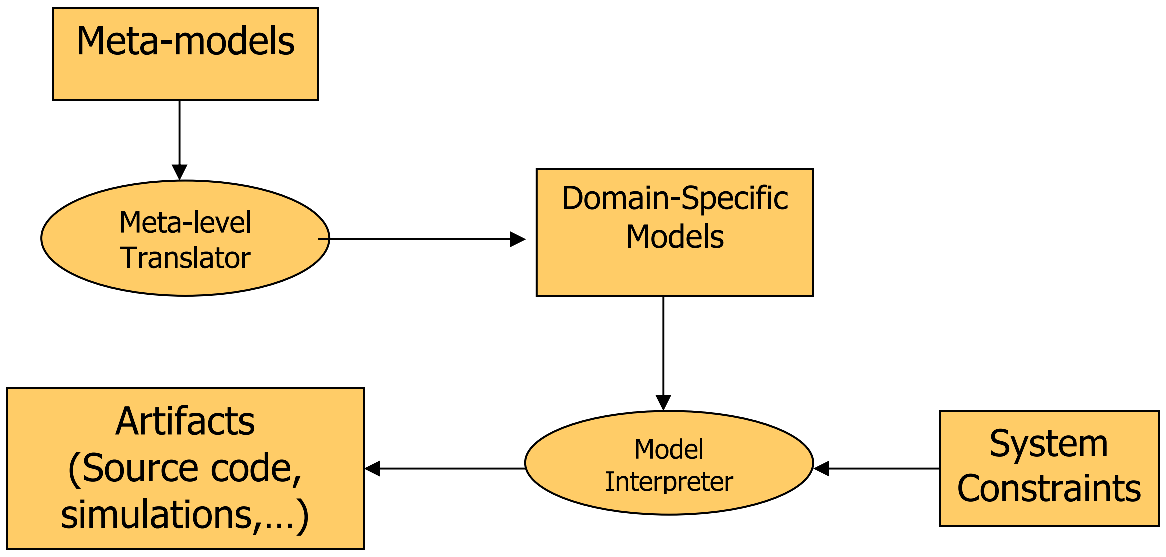

Figure 1: Overview of Model Integrated Computing Process

systems. With MIC, a modeling environment operates according to a modeling paradigm, which is a set of requirements that govern how a system within a domain is to be modeled. The modeling paradigm is captured in the form of formal modeling language specifications called a meta-model. As shown in Figure 1, once a meta-model is created for a particular domain, a modeling environment is constructed (through meta-level translation) that allows a modeler to create domain-specific models that can be synthesized into various artifacts. The Generic Modeling Environment is a meta-programmable tool that implements the ideas of MIC [9]. The GME has been successfully used on dozens of research projects representing numerous domains (avionics, automotive, electrical utilities, chemical plants, and numerous military projects – see http://www.isis.vanderbilt.edu).

In most model-based approaches, like MIC and the MDA, a base model is often created that is independent of several implementations. In our approach to MIC, this base model is augmented with various system constraints that refine the model into a more concrete representation. There is a fundamental problem, however, in the practical introduction of constraints into a base model. It is often difficult to specify and manage constraints that crosscut the model [5]. That is, the insertion of a global system constraint often requires a modeler to visit and change *multipl*e locations within the model, or across the model. This can be a very difficult task for anything but a simple model (see Figure 2). A solution technique for handling this problem is presented in the next section.

The MIC approach provides the *first level of independence* in the CoSMIC (Component Synthesis using Model-Integrated Computing) [11] solution to MDA. Through meta-modeling and generative programming techniques [1], it permits the construction of models for any domain as well as its synthesis to any underlying platform (a modeler must still write the associated interpreters to provide the platform target generation, however).

3. Aspect-Oriented Domain Modeling

As an initial solution to the problem of modeling crosscutting modeling concerns, an aspect-oriented approach [8] to domain-modeling has been adopted. In our current implementation, a special language has been created to describe the crosscutting features of the model. Initially, there was a lack of tool support for automatically weaving constraints into model-based systems. We have constructed a model weaver to accomplish the task of providing proper modularization of crosscutting modeling concerns [5]. This permits a modeler to more easily make changes to the base model without manually visiting multiple locations in the model. Thus, the weaver and its associated language permit the modeler to make statements of quantification across the model (an important part of aspect-orientation, as noted in [3]).

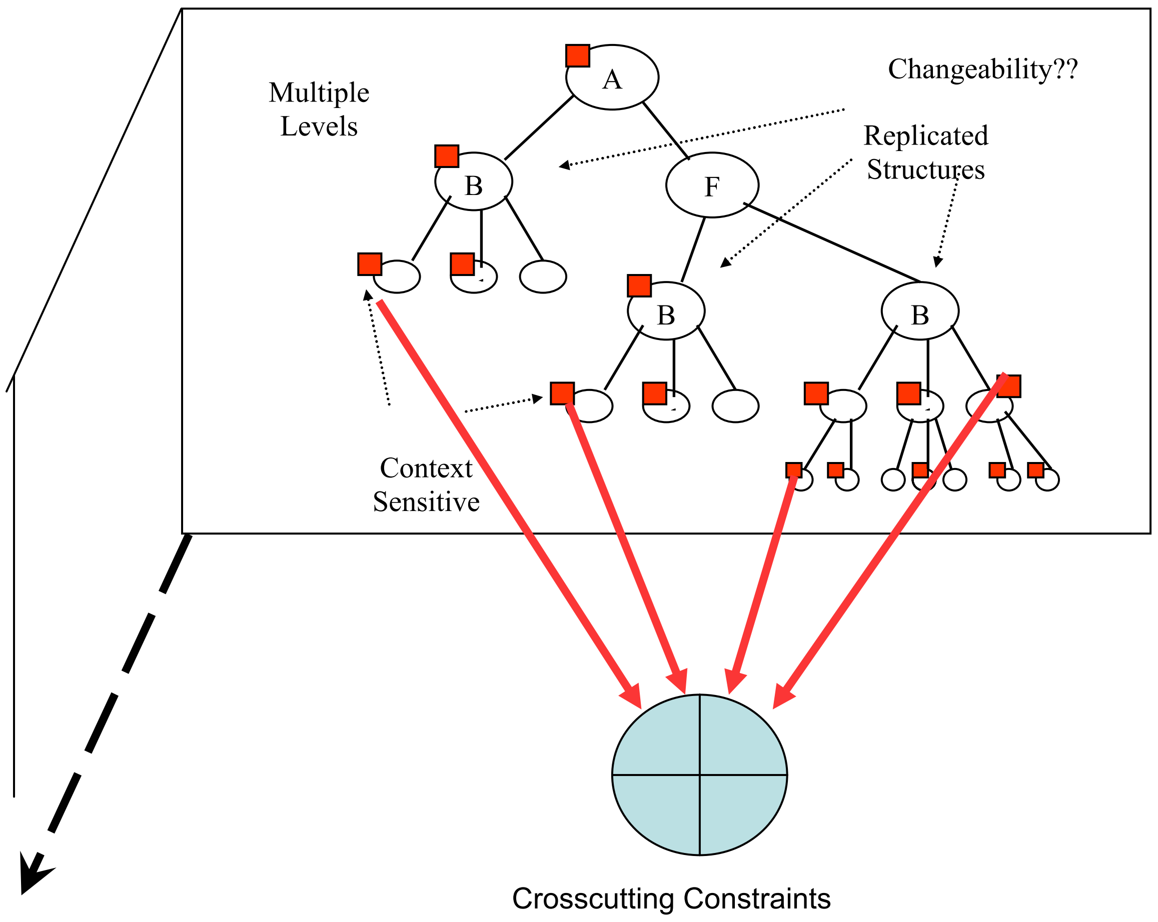

Figure 2: The Multi-dimensions of Crosscutting Model Aspects

Our approach to model weaving is depicted in Figure 3. In this figure, constraint-free base models serve as an input to the weaver, and the output of the weaver is a new model that has the constraints dispersed across the original base. To perform this process, specification aspects are used to denote those locations in the model where a crosscutting constraint is to be applied. A strategy is a general heuristic for performing the transformation that is needed to properly insert the constraint into a given context.

There have been other approaches to applying aspect-orientation to higher levels of abstraction. For example, an idea for aspect-oriented requirements specification is presented in [10]. Additionally, there are several workshops that have been conducted on the topic of aspect-oriented modeling. However, these previous efforts do not provide the type of tool support that is needed to realize an MDA-based solution.

This approach to weaving as a supplementation to MDA is a key part of the CoSMIC toolsuite that is being constructed in our collaboration with Vanderbilt University [11]. This idea of Aspect-Oriented Domain Modeling (AODM) represents the *second level of independence* by permitting the separation of the base model from a concretizing set of system constraints that are used to drive the platform dependent code generation. (Note: Here, platform dependence does not always imply the generation of targeted middleware variations. It often means, as in the case of modeling embedded systems, the notion of platform representing specific hardware configurations.)

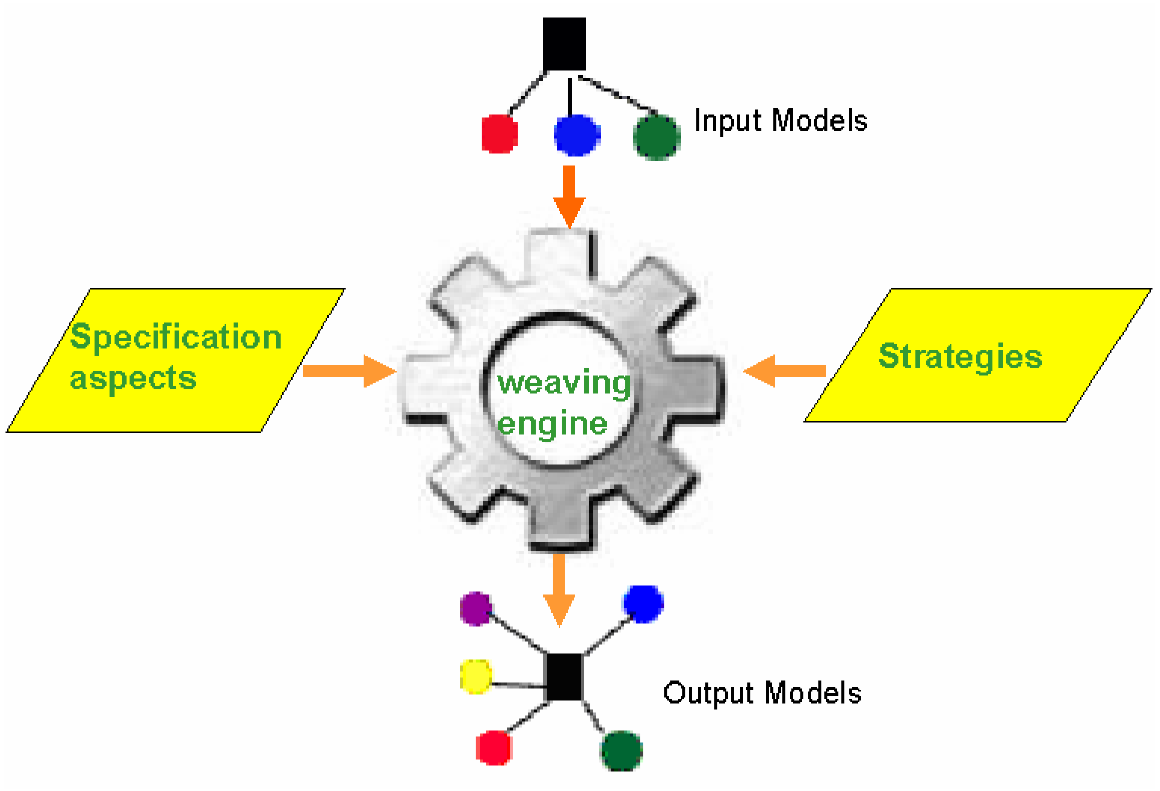

Figure 3: Effect of Domain-Specific Weaving

4. Tool-Independent Model Weaving

We are currently working on a new focus for applying our ideas of model weaving. The previous two sections describe levels of independence for supporting variation within a particular domain, and a specific targeted platform. In this section, we describe yet another level of independence for concepts of model weaving. This *third level of independence* addresses the need for a core weaving engine that is independent of modeling environments.

Our current implementation of the model weaver is specifically tied to the GME modeling tool. Our next goal is to construct the core of our weaving engine such that it can be adapted and used with different modeling tools. For example, the core of the weaving engine could be adapted to work with other modeling tools, such as Rational Rose, or new environments like Cadena [6] (this is a modeling tool for the CCM that is based upon the Eclipse environment).

To construct a tool-independent weaver, we are focusing our efforts on two primary components: 1) a core weaving engine, and 2) the associated adapters that are needed to wrap the engine around the exposed APIs for accessing the model data structures of each tool. The core weaving engine can be adapted to multiple modeling environments to weave cross-cutting constraints to domain models. The adapters are used to integrate the core weaving engine to different modeling tools. Figure 4 depicts this process.

5. Conclusion

Aspect-Oriented weaving of domain models is a technique that combines the ideas of MIC [9] and AOP [8]. The main benefit is to facilitate rapid construction and evolution of domain models via flexible weaving of cross-cutting constraints [5]. We are applying this idea to support the basic tenets of MDA, whereby platform specific attributes are separated from a base model.

In this paper, we described three different levels for achieving independence within a specific approach to MDA. That is, the MIC/GME tools provide the needed level of domain independence; the aspect weavers are a great help for separating platform specific details; and the future goal supports a third level that is concerned with tool-independence.

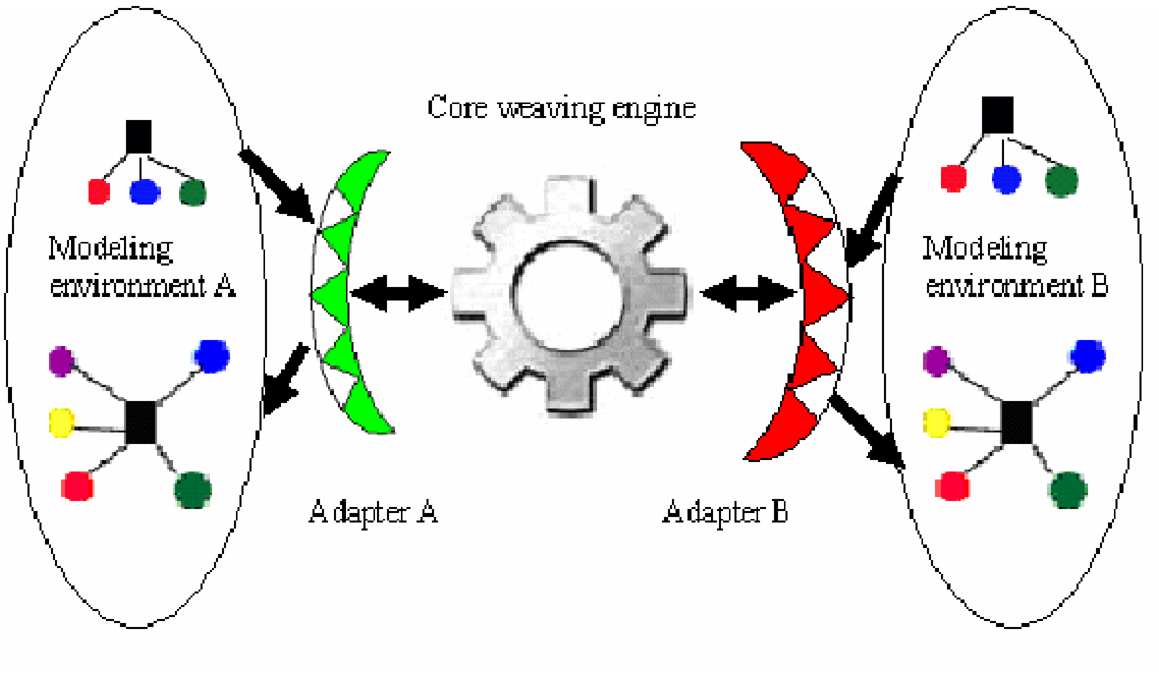

Figure 4: Core Weaving Engine Adapted to Different Modeling Environments

We are beginning a detailed effort to model and synthesize CCM using the GME. At the workshop, we would like to demonstrate our tools by weaving various modeling aspects into a base model. After weaving in a set of crosscutting modeling concerns, our demo will show the generation of CIAO [14] and FACET [7] code from the transformed models, as applied to Boeing's BoldStroke framework [12].

Acknowledgement

This research is funded by the DARPA Information Exploitation Office (DARPA/IXO), under the Program Composition for Embedded Systems (PCES) program

References

[1] Krzysztof Czarnecki and Ulrich Eiseneker, *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, 2000.

[2] E. W. Dijkstra, "Notes on Structured Programming: On Program Families," *Structured Programming*, Academic Press, London, 1972, pp. 39-41.

[3] Robert Filman and Dan Friedman, "Aspect-Oriented Programming is Quantification and Obliviousness," *OOPSLA Workshop on Advanced Separation of Concerns*, Minneapolis, Minnesota, October 2000.

[4] David S. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*, John Wiley and Sons, 2003.

[5] Jeff Gray, Ted Bapty, Sandeep Neema, and James Tuck, "Handling Crosscutting Constraints in Domain-Specific Modeling," *Communications of the ACM*, October 2001, pp. 87-93.

[6] John Hatcliff, William Deng, Matthew Dwyer, Georg Jung, Venkatesh Prasad, "Cadena: An Integrated Development, Analysis, and Verification Environment for Component-based Systems," To appear in Proceedings of the *International Conference on Software Engineering*, Portland, OR, May 2003.

[7] Frank Hunleth, Ron Cytron, and Chris Gill, "Building Customized Middleware Using Aspect-Oriented Programming," *OOPSLA Workshop on Advanced Separation of Concerns*, Tampa, Florida, October 2001.

[8] Gregor Kiczales, Eric Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William Griswold, "Getting Started with AspectJ," *Communications of the ACM*, October 2001, pp. 59-65.

[9] Ákos Lédeczi, Arpad Bakay, Miklos Maroti, Peter Volgyesi, Greg Nordstrom, Jonathan Sprinkle, and Gábor Karsai, "Composing Domain-Specific Design Environments," *IEEE Computer*, November 2001, pp. 44-51.

[10] Awais Rashid, Ana Moreira, Joao Araujo, "Modularization and Composition of Aspectual Requirements," *Conference on Aspect-Oriented Software Development*, Boston, MA, March 2003.

[11] Douglas C. Schmidt, Aniruddha Gokhale, Balachandran Natarajan, Sandeep Neema, Ted Bapty, Jeff Parsons, Andrey Nechipurenko, Jeff Gray, and Nanbor Wang, "CoSMIC: A MDA tool for Component Middleware-based Distributed Real-time and Embedded Applications," *OOPSLA Workshop on Generative Techniques for Model-Driven Architecture*, Seattle, WA, November 5, 2002.

[12] David Sharp, "Reducing Avionics Software Cost Through Component Based Product-Line Development," *Software Technology Conference*, Salt Lake City, Utah, April 1998.

[13] Gerald Jay Sussman, "Robust Design through Diversity," DARPA Amorphous Computing Workshop, 1999.

[14] Nanbor Wang, Krishnakumar Balasubramanian, and Chris Gill, "Towards a real-time CORBA Component Model," in *OMG Workshop On Embedded & Real-Time Distributed Object Systems*, Washington, D.C., July 2002, Object Management Group.

[15] Paul Ward and Stephen Mellor. *Structured Development for Real-Time Systems* Yourdon Press, 1985.