Handling Crosscutting Constraints in Domain-Specific Modeling

Jeff Gray, Ted Bapty, Sandeep Neema, James Tuck

Institute for Software Integrated Systems (ISIS)
Vanderbilt University, Nashville TN 37235
{jgray, bapty, neemask, tuckjm}@vuse.vanderbilt.edu
http://www.isis.vanderbilt.edu

ABSTRACT

Domain-specific models for embedded systems often contain constraints that aid in stipulating design criteria. These constraints, however, are typically scattered across a model hierarchy in such a manner that it is difficult to reason about the effect and purpose of each constraint. This poster describes an approach for providing better separation of concerns with respect to constraints.

1. PROBLEM BACKGROUND

Our core research area at ISIS is model-integrated computing (MIC) [6]. A major focus of MIC is on domain-specific modeling environments that are created from metalevel specifications of a particular domain. A metaCASE tool called the Generic Modeling Environment (GME) has been developed for this purpose.

In many real-time embedded systems it is advantageous to model the design space of an application. In fact, this is mandatory for self-adaptive systems that must chose at run-time among numerous alternatives. Several of the domain models that we have created are embedded real-time systems that are highly adaptive. Our approach to modeling self-adaptive embedded systems uses a form of OCL [8] constraints to help prune the size of the design space during exploration. These constraints stipulate design criteria and limit design alternatives [5].

Unfortunately, such constraints are tangled throughout the model hierarchy. These constraints cut across the modular boundaries of a model. The crosscutting nature of these constraints makes it difficult to maintain and reason about their effects and purpose.

The specific details of the work presented in this poster can be found in [3]. In this poster presentation we seek a forum for engaging in one-on-one discussions that were not possible with the initial description of our work.

2. APPROACH

Our solution to the problem of tangled constraints involves the separation of constraints from modeling elements. The solution allows modular specifications of constraints to be propagated throughout a model via a domain-specific weaver. The purpose of a weaver is to integrate constraints back into a model. In general, the solution is an extension to research in Aspect-Oriented Programming (AOP) [4] and is based on the concepts of generative programming [1].

Domain-specific weavers rely on specification aspects and strategies to carry out their duty. ***Specification aspects***, similar to pointcuts in AspectJ [4], are used to specify where the constraints will be applied in the model. ***Strategies*** describe how a constraint is applied in the context of a particular node in the model. The description of specification aspects and strategies allows a modeler to quantify properties of the model in a module that is separate from the model structure.

Domain-specific weavers are created as a particular instantiation of a metaweaver framework. A core component of this framework is a code generator that translates high-level descriptions of strategies, specified as a domain-specific language (DSL), into C++ source code. We call this DSL the Embedded Constraint Language (ECL). It is based on the OCL [8].

3. FUTURE RESEARCH FOCUS

Extensions to the initial domain-specific weaver framework are continuing to be developed. For instance, although strategies allow for variability among different GME paradigms, there are other improvements that can be made to the framework in order to extend its variability. Each extension requires a new DSL and generator.

The input to a weaver built with our framework assumes that the separation of constraints is being performed on models created with the GME and exported as an XML file. The limitation imposed by this assumption precludes other modeling tools (that can also export models using XML) from being able to employ the benefits of a constraint weaver. A goal of this research will be to demonstrate that additional variability can be achieved by generating pieces of the XML parser from the underlying DTD of the modeling tool.

As noted previously, the metaweaver framework for domain-specific modeling uses the ECL for expressing both strategies

and specification aspects. A point of variation within the framework is an extension that would allow the specification aspect parser to be replaced with some other language. To provide variation with respect to the aspect parser, the output of a parser generator (e.g., YACC or PCCTS) needs to be integrated into the framework. It is also uncertain at this point whether a framework that provides this level of variability needs the capabilities of strategies. In place of strategies, it may be the case that a traversal/visitor language is needed [7].

Building on the ideas of extension just described, we are developing a weaver framework that will make it easier to mix and match different base languages (e.g., Ada, Delphi, Prolog) with various aspect languages. The details of this extension were first presented in [2].

4. ACKNOWLEDGEMENT

This work has been supported by the DARPA Information Technology Office (DARPA/ITO), under the Program Composition for Embedded Systems (PCES) program, Contract Number: F33615-00-C-1695.

5. REFERENCES

1. Czarnecki, K., and Eiseneker, U., *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, 2000.
2. Gray, J., "Using Software Component Generators to Construct a MetaWeaver Framework," *International Conference on Software Engineering (ICSE)*, Doctoral Symposium, Toronto, Ontario, Canada, May 2001.
3. Gray, J., Bapty, T., Neema, S, and Tuck, J., "Handling Crosscutting Constraints in Domain-Specific Modeling," *Communications of the ACM*, October 2001.
4. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W., "Getting Started with AspectJ," *Communications of the ACM*, October 2001.
5. Neema, S., and Ledeczi, A., "Constraint Guided Self-Adaptation," *International Workshop on Self-Adaptive Software*, Balatonfured, Hungary, May 2001.
6. Nordstrom, G., Sztipanovits, J., Karsai, G., and Ledeczi, A., "Metamodeling - Rapid Design and Evolution of Domain-Specific Modeling Environments," *IEEE ECBS Conference*, Nashville, TN, April 1999.
7. Ovlinger, J., and Wand, M., "A Language for Specifying Recursive Traversals of Object Structures," *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, Denver, CO, November 1999, pp. 70-81.
8. Warmer, J., and Kleppe, A., *The Object Constraint Language: Precise Modeling with UML*, Addison-Wesley, 1999.