A Framework for Creating Aspect Weavers

Jeff Gray

Institute for Software Integrated Systems (ISIS)
Vanderbilt University, Nashville TN 37235
jgray@vuse.vanderbilt.edu
http://www.vuse.vanderbilt.edu/~jgray

ABSTRACT

Several new modularity technologies have been proposed that improve separation of concerns in programming languages. In particular, research in Aspect-Oriented Programming (AOP) has been promoted as a means toward the separation of concerns that crosscut the modularity of an implementation. This brief summary presents two research objectives for extending AOP. First, the concept of AOP is investigated at a higher level of abstraction. A core focus of this objective is the application of aspect-oriented (AO) techniques to model-integrated computing. The second research objective involves the creation of a framework that aids in the construction of new aspect weavers. The framework utilizes several domain-specific languages (DSLs) and generators to provide variability among weaver instances.

1. INTRODUCTION

Even though separation of concerns is an old idea, one can witness the nascence of a research area devoted to exploring new techniques to support advanced separation of concerns [7]. In AOP, the focus is on capturing, in a modular way, the crosscutting concerns of a system [5]. AOP recognizes that crosscuts are inherent in most systems and are generally not random. An *aspect*, therefore, is a piece of code that describes a recurring property of a program that crosscuts the system. The goal of AOP is to support the programmer in cleanly separating components and aspects from each other by providing new language constructs that make it possible to abstract and compose them to produce an overall system.

Our core research area at ISIS is model-integrated computing [6]. This work is focused on domain-specific modeling environments that are created from metalevel specifications of a particular domain. Several of the domain models that we have created are embedded real-time systems that are highly adaptive. These models contain constraints that stipulate design criteria and limit design alternatives. Such constraints are tangled throughout the hierarchy. The crosscutting nature of these constraints makes it difficult to maintain and reason about their effects and purpose.

In AOP, a translator called a weaver is responsible for taking code specified in a traditional programming language and additional code specified in an aspect language, and weaving the concerns together. We are uniting our core research area with the powerful new techniques offered in AOP by extending the purview of applicability by developing weavers for constraints in domain-specific models.

2. GOALS

The goals of this thesis can be summarized by two research objectives:

- Raise the concepts of AO to a higher level of abstraction

  An Aspect-Oriented (AO) approach can be beneficial at different stages of the software lifecycle and at various levels of abstraction. Whenever the description of a software artifact exhibits crosscutting structure, the principles of modularity espoused by AO offer a powerful technology for supporting separation of concerns. This thesis makes a novel contribution to the literature on advanced separation of concerns by investigating the application of AO techniques to domain-specific modeling [3].

- Assist in the creation of new weavers using a metaweaver framework

  Because the syntax and semantics of each modeling domain are dissimilar, a different weaver is needed for each domain. A metaweaver framework is proposed as an aid toward constructing new domain-specific weavers. This framework will make use of several code generators that take metalevel specifications (described in a DSL) as input and produce code that will serve as a hook into the framework.

  Additionally, the initial domain-specific metaweaver framework already developed will undergo several modifications so that a weaver for programming and aspect languages other than Java and AspectJ can be constructed. Initial results for this research objective are described in [2].

3. APPROACH

The specific details of the approach for applying AO techniques to domain-specific models are covered in [3]. One of the key elements of the approach is the application of a DSL (we have developed an extension to OCL [8]) that is used to navigate the domain model and to quantify the location of specific model constraints. The same DSL is used to specify strategies that implement the computations and propagations of constraints that are peculiar to a particular domain.

A generative programming approach has been adopted such that the DSL is processed by a generator that produces C++ code [1]. This C++ code is used to implement the domain-specific strategy and is linked into a weaver framework.

From experience in other work [4], we have found that a framework that uses software generators greatly reduces the amount of time needed to create new applications. The concepts that are used in the framework for creating domain-specific weavers will be extended to offer support in the creation of weavers for programming and aspect languages. More concrete details of this concept are described in [2].

4. STATUS AND FUTURE

The framework for creating domain-specific weavers has been under development for over a year. Instantiations of the weaver have been used to assist in the quantification of constraints in two different domains. The weavers were used to apply constraints that focused on power consumption and processor assignment in models that represent real-time embedded systems. Future extensions are planned to permit more powerful strategies to be specified in the DSL.

The weavers created from this framework currently accept as input models that have been stored using a specific XML DTD. We plan to expand on this work by creating another generator that will provide variability with respect to the XML format that is understood by a weaver.

The work on extending the framework to include weavers for programming and aspect languages is still in an immature state. The continued development of these enhancements is the focus of the remaining work needed to accomplish the research objectives of this thesis.

5. 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.

6. 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 Meta-Weaver Framework," *23rd International Conference on Software Engineering (ICSE 2001),* 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. Karsai, G., and Gray, J., "Component Generation Technology for Semantic Tool Integration," *IEEE Aerospace Conference*, Big Sky, MT, March 2000.
5. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W., "Getting Started with AspectJ," *Communications of the ACM*, October 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. Tarr, P., Ossher, H., Harrison, W., and Sutton, S., "N Degrees of Separation: Multi-Dimensional Separation of Concerns," *International Conference on Software Engineering (ICSE)*, Los Angeles, CA, May 1999.
8. Warmer, J., and Kleppe, A., *The Object Constraint Language: Precise Modeling with UML*, Addison-Wesley, 1999.