Using Software Component Generators to Construct a Meta-Weaver Framework[1]

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. The initial efforts to demonstrate these technologies are usually focused on a single programming language. Since we live in a polyglot world, this proposal addresses the goal of being able to take these new powerful technologies to other languages. The approach uses software generators that create new "weavers" from meta-specifications of programming languages.

1. Introduction

Aspect-Oriented Programming (AOP) provides a strategy for dealing with concerns that crosscut modularity [1]. Crosscutting results in tangled code that is often redundant and difficult to reason about and change. A goal of AOP is to provide new language constructs that allow a better separation of concerns for these crosscutting aspects. For a detailed example of how AOP was used to improve modularity, see [2].

In AOP, a translator called a weaver is responsible for taking code specified in a traditional (base) programming language, and additional code specified in an aspect language, and merging the two together. Because the aspect code describes numerous behaviors that crosscut a system, the concerns must eventually be integrated into the base code. This is the purpose of a weaver. The most mature AOP weaver is bound to Java and a generic aspect language called AspectJ (see http://aspectj.org)

Obviously, the benefits espoused by AOP will be desired by those who use programming languages other than Java, or those who want to use an aspect language other than the one provided in AspectJ. This proposal addresses the problem of creating new weavers for these other languages.

2. Proposed Solution

A weaver framework is proposed that will support the concept of a meta-weaver, or weaver-compiler, that creates new weavers from the meta-specifications of a base language and an aspect language. Thus, once the initial meta-specifications are provided for all of the base and aspect languages, we could have an Ada, Prolog, or even PL/SQL version of a weaver for several different aspect languages that we have also defined. In order to build a new weaver using this approach, the following must be provided to the framework:

1. A meta-level description of the key elements of the base language is needed, as well as a description of the relations between each element (e.g., classes contain methods and attributes). In a previous project, we have developed a meta-specification language to accomplish this [3]. The same meta-level description is also needed for the aspect language. Each meta-level description is fed into a generator that creates C++ code representing the description.
2. The weaver must know how to parse the base programming language. Therefore, this must be described using a parser generator like PCCTS. The weaver must also know how to parse the aspect language. The parser will build a parse tree based on the code generated from the meta-level language descriptions.
3. An interpreter must be able to walk the generated parse trees of the base and aspect languages and be able to integrate the crosscutting concerns. In a past project we also have gained experience in specifying higher-level traversal/visitor sequences [3]. An adaptation of this method could be useful here and would build on the work of [4] and [5].

Note that the framework depends on three different types of code generators. Each base and aspect language becomes componentized into each new weaver that is generated.

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

3. Related Work

Several researchers are working in the area of Multi-Dimensional Separation Of Concerns (MDSOC) to provide new language constructs to handle crosscutting; see, for example, Subject Oriented Programming (SOP) [6]. A recent book provides a detailed discussion of the issues involved in this area [7].

Perhaps the work that is closest in intent to this proposal can be found in [8]. The Jakarta Tool Suite (JTS) contains the basic tools to support the addition of new programming features to the Java language. It assists in the construction of new preprocessors for domain specific languages (DSLs) that are transformed into a host language. The supported host language in JTS is called Jak. Jak is described as a superset of Java that supports meta-programming. It seems likely that JTS could be used to create a weaver for new aspect languages to support Java. This work differs from our approach in that it is fixed to a superset of Java (Jak). We are interested in not only extending the DSL, or aspect language, but also the base programming language.

4. Status and Future

As an exercise to show the feasibility of the approach, a weaver for subsets of two core and aspect languages will be created. The meta-weaver will also be applied at new levels of abstraction. For example, in the types of models that we build at ISIS/Vanderbilt to support our research, constraints represent a type of crosscutting concern. The meta-weaver framework will be used to create a weaver for handling these models and constraints. This will apply the principles of aspect-orientation to a higher level of abstraction; from programming to modeling [9]. In fact, a primitive version of a weaver to support these crosscutting model constraints has already been created. It allows different weavers to be generated based on specifications of the domain represented in our reconfigurable modeling environment [10].

5. Summary

From experience in other work [3], we have found that a framework that uses software generators greatly reduces the amount of time needed to create new applications. The meta-specification for a particular language permits the language itself to be treated as a kind of component that can be plugged into the framework. Inserting the meta-specifications into the weaver framework can create new weavers for different languages.

A meta-weaver can offer the distinguishing advantage of being able to take the concepts of AOP to new base and aspect languages. The power of AOP would be available to many new developers who use languages other than Java and AspectJ. In fact, the meta-weaver concept can be positioned as a tool to assist in areas outside of AOP. This proposal actually applies to any situation where a base programming language is extended with a new type of DSL.

References

1. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J. Loingtier, and J. Irwin, "Aspect-Oriented Programming," *European Conference on Object-Oriented Programming* (ECOOP '97), Finland, 1997.
2. M. Lippert and C. V. Lopes, "A Study on Exception Detection and Handling Using Aspect-Oriented Programming," *International Conference on Software Engineering* (ICSE 2000), Limmerick, Ireland. 2000.
3. G. Karsai and J. Gray, "Component Generation Technology for Semantic Tool Integration," *IEEE Aerospace Conference*, Big Sky, MT, 2000.
4. J. Ovlinger and M. Wand, "A Language for Specifying Recursive Traversals of Object Structures," *Conference on Object-Oriented Programming, Systems, Languages, and Applications* (OOPSLA '99), Denver, CO, 1999.
5. K. Lieberherr, *Adaptive Object-Oriented Software*, International Thomson Publishing, 1996.
6. P. Tarr, H. Ossher, W. Harrison, and S. Sutton, "N Degrees of Separation: Multi-Dimensional Separation of Concerns," *International Conference on Software Engineering* (ICSE '99), Los Angeles, CA, 1999.
7. K. Czarnecki and U. Eiseneker, *Generative Programming: Methods, Tools, and Applications*, Addison-Wesley, 2000.
8. D. Batory, B. Lofaso, and Y. Smaragdakis, "JTS: Tools for Implementing Domain-Specific Languages," *Fifth International Conference on Software Reuse*, Victoria, Canada, 1998.
9. J. Gray, T. Bapty, and S. Neema, "Aspectifying Constraints in Model-Integrated Computing," *OOPSLA 2000: Workshop on Advanced Separation of Concerns*, Minneapolis, MN, 2000.
10. G. Karsai, G. Nordstrom, A. Ledeczi, and J. Sztipanovits, "Specifying Graphical Modeling Systems Using Constraint-based Meta-models," *IEEE Symposium on Computer Aided Control System Design*, Anchorage, Alaska, 2000.