An Aspect-Oriented Infrastructure for a Typed, Stack-based, Intermediate Assembly Language

Douglas R. Dechow
Dearborn 311
Oregon State University
Corvallis, OR 97331
541-737-4145
dechow@cs.orst.edu

ABSTRACT

While traditional, one-dimensional approaches to the problem of separation of concerns have been adequate for current software development, they are often brittle and resistant to evolutionary change. Aspects and aspect-orientation offer a controllable, modular mechanism for describing the separation of concerns that are orthogonal to the object model that is the primary developmental focus of a wide range of software applications. This dissertation research project involves the creation of an aspect-oriented infrastructure to support a variety of software development tools. Use of this infrastructure is demonstrated in domain areas such as ecological modeling software and web development in order to establish aspect-orientation as a feasible and straightforward solution to the problem of separation of concerns in object-oriented software systems. In the process of establishing the viability of the aspect-oriented solution, this dissertation investigates several new directions in aspect-orientation: aspects in system software, language independent aspects, aspect integration techniques, and opportunities for aspect reuse. In comparing the two-dimensional, aspect-oriented approach to the traditional, one-dimensional approach, the assertion of this research is that a two-dimensional approach offers an inherently more flexible software system while maintaining the advantages of modularity and code reuse that have long been ascribed to object-oriented systems.

Keywords: Separation of concerns, common intermediate language (CIL), preprocessors, aspects, aspect-oriented programming, aspect weaver, compile-time, runtime, weave-time, components, ecological modeling.

1. Rationale

A distinct problem associated with introducing a new software simulation system to the world of ecological modeling is that many would-be users are not proficient programmers. Mechanisms such as tracing and logging can be used to provide feedback to model developers. Additionally, the ability to monitor and evaluate the state variables as the simulation evolves would be useful. However, such mechanisms are not salient features of the ecological model itself. As such, they can be seen as orthogonal to the model and a potential source of tangled code. Using aspect-oriented approaches to automate the control of these concerns can remove this burden from the modeler while simultaneously making the features available to provide the necessary feedback.

ModCom is a component-based simulation tool for distributed ecological modeling [5]. ModCom is an object-oriented application framework (C++), and it is built on the Microsoft Component Object Model (COM). Although a visual development environment for ModCom is under development, current users of the framework must write their simulations in C++ or Visual Basic (VB developers are, potentially, a large part of ModCom's user base).

An aspect-oriented infrastructure to interoperate with ModCom is currently under development. One target of the AO infrastructure is the Rotor implementation of the .NET Common Language Runtime (CLR).

Most aspect-weavers operate at compile time (AspectJ)[2] or runtime (AOP/ST, AspectS) [1][3]. The aspect weaver that is being developed for ModCom will operate in the region between the two.

In the strictest sense, the weaver will be a compile time weaver. However, by making use of multiple translation model and the runtime facilities (specifically the class loader) of the CLR, it will appear to the user as if the weaving of aspects is occurring at runtime. For the purposes of this research, this intermediate time frame will be referred to as *weave-time*.

2. Objectives

The goal of this project is to design and implement an aspect-oriented infrastructure that interoperates with the ModCom ecological simulation tool. Presently, the only way for a model developer to obtain feedback while creating a simulation is via the tried and true method of littering their source code with "print" statements.

Instead of forcing model developers to tangle their model's source code in an ad-hoc fashion, the aspect-oriented infrastructure can be used to instrument a simulation in a controlled manner. Separating the feedback/instrumentation concern from the model concern will allow inexperienced programmers to focus on building simulations.

Visualization is another area of ModCom that is amenable to treatment via the aspect-oriented infrastructure. Currently, visualization in ModCom is handled by COM graphing and charting components.

Additionally, the project will offer opportunities to evaluate potential aspect reuse techniques in the form of the

OOPSLA '02, November 4-8, 2002, Seattle, Washington, USA.
ACM 1-58113-626-9/02/0011.

tracing and logging aspect libraries. Given the inherent multi-language support that is present in the CLR, the development of multi-language aspect libraries will be examined.

Initially, the needs of the ModCom user base require that the OO CLR languages—C#, VB, VC++--will be targeted. Support for the non-OO CLR languages will be investigated as the project moves forward.

3. Approach

The core of an aspect-oriented infrastructure rests upon the system's ability to accomplish three tasks: "a join point model, a means of identifying join points, and a means of affecting implementation at join points."[4] The three key features of the research project that provide this functionality are outlined below.

3.1 Aspect weaver

The aspect weaver will be a source-to-source preprocessor designed to manipulate the Common Intermediate Language (CIL) of the .NET CLR. In the final paragraph of the rationale section, it was mentioned that the CLR supports a multiple translation model. The high-level language compilers of the CLR all emit CIL. This is the first compilation/translation step. Next, the CIL is compiled into native machine code by the CLR's JIT compiler. Weave-time is the time frame that exists between the translation of a high-level language to CIL and the secondary JIT compilation of the CIL into machine code. In the proposed system, weave-time takes place under the direction of the user.

3.2 Aspect Libraries

An integral part of this project is identifying and developing the aspects themselves. Initially, this effort will focus on the creation of aspects that can aid the users of ModCom in developing simulations. Eventually, aspect libraries for tracing and logging will be developed. The use of these aspects in other contexts will be investigated.

3.3 Aspect Integrator

A visual aspect integration tool will be a modeler's primary means of interacting with the infrastructure. The tool will allow the users to select join points in their models and aspects from the libraries. The visual tool will be designed so that the user will have no explicit knowledge that they are weaving aspects into the model.

4. ACKNOWLEDGMENTS

I would like to thank Dr. Tim Budd for serving as my advisor for this dissertation. I would also like to thank Dr. Budd and Dr. John Bolte, a professor in the Bioresource Engineering Department at Oregon StateUniversity, for supporting this work under an EPA grant entitled "Developing Object-based Simulation Tools for Distributed Modular Ecological Modeling".

5. REFERENCES

[1] K. Boellert. The AOP/ST homepage, http://www.theoinf.tu-ilmenau.de/~kaib/aop/.

[2] The AspectJ homepage, http://www.aspectj.org.

[3] R. Hirschfeld. AspectS—AOP with Squeak. *OOPSLA 2001 Workshop on Advanced Separation of Concerns in Object-Oriented Systems*. 2001.

[4] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W.G. Griswold. Getting Started with AspectJ. *Communications of the ACM*, 44(10):59-65, October 2001.

[5] The ModCom homepage, http://biosys.bre.orst.edu/ModCom/.