C-SAW and GenAWeave: A Two-Level Aspect Weaving Toolsuite

Jeff Gray, Jing Zhang, Suman Roychoudhury
Department of Computer and Information Sciences
University of Alabama at Birmingham
Birmingham, AL 35294 USA
1-205-934-8643
{gray, zhangj, roychous} @ cis.uab.edu

Ira Baxter
Semantic Designs, Inc.
Austin, TX 78759 USA
1-512-250-1018
idbaxter@semdesigns.com

ABSTRACT
This demonstration will feature overviews of the C-SAW and GenAWeave projects. The first half of the presentation will introduce the concept of two-level aspect weaving, which unites a *model transformation* tool with a *program transformation* engine. From models representing an avionics application, it will be shown how changes to model properties trigger corresponding adaptations to the related source code. The second half of the demonstration is focused on using a program transformation engine to perform the task of aspect weaving. In particular, several crosscutting concerns will be woven into an Object Pascal application.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques, D.2.6 [**Software Engineering**]: Programming Environments – *graphical environments* and F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems.

General Terms
Design, Experimentation, Languages.

Keywords
Model-Driven Architecture, Aspect-Oriented Programming, Program Transformation, Software Maintenance.

1. INTRODUCTION
The C-SAW and GenAWeave tools support evolution of object-oriented legacy software through a two-level approach using aspects [6]. The principal strategy of these tools is to generate low-level transformation rules from higher-level domain languages. The generated transformation rules, along with the initial version of the application source code, serve as input to the Design Maintenance System (DMS) from Semantic Designs [1]. The generated rules drive the transformation process in order to produce a modified version of the source containing new concerns that have been woven across the application code base. The demonstration will show the ability to make rapid adaptations to a large cross-section of an application through simple specification changes at a high-level of abstraction.

As case studies, the demonstration will highlight the transformation of two legacy commercial applications: a large mission-computing avionics framework written in C++, and a client-server enterprise management system implemented in Object Pascal. In the avionics application, transformation rules are generated from domain-specific models created in the Generic Modeling Environment (from Vanderbilt University) [4]. Using C-SAW [3], it will be shown that small changes in a representative model can regulate concurrency and logging policies across many C++ classes. The Object Pascal portion of the demonstration will illustrate the use of DMS as the underlying engine for an aspect weaver. A unique feature of the demonstration is the ability to weave aspects into a legacy language (not Java) at the source level using GenAWeave. The remaining sections summarize C-SAW and GenaWeave.

2. MODEL-DRIVEN TRANSFORMATION
The first half of the demonstration will show the feasibility of utilizing the capabilities of a program transformation system to support parsing and source-level transformation of legacy code. In particular, the C-SAW model transformation tool will be united with DMS to enforce a *causal connection* between models and corresponding legacy source through an approach that we call *model-driven program transformation* (MDPT). A causal connection occurs whenever any modifications are made to the models, such that a corresponding change is applied to the source.

Figure 1 illustrates an overview of the MDPT concept. In our approach, DMS is utilized as the underlying program transformation mechanism. The core component of DMS is a term rewriting engine that supports the powerful capabilities for pattern matching and source translation. In the MDPT idea, domain-specific model interpreters are constructed to generate the DMS transformation rules from the evolving features of the instance models. The corresponding legacy source code, along with the transformation rules, will be fed into DMS. As a result, the legacy source is transformed as the generated rules are applied by DMS.

OOPSLA'04, Oct. 24–28, 2004, Vancouver, British Columbia, Canada.
ACM 1-58113-833-4/04/0010.

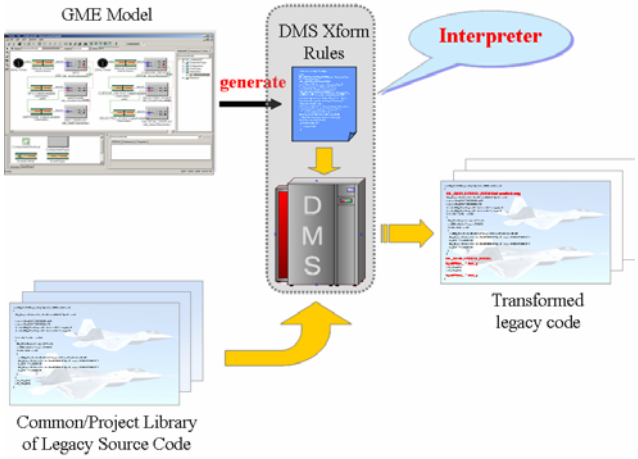

Figure 1. Model-Driven Program Transformation

It is worth noting that the domain experts are not supposed to understand the DMS rules. Domain experts modify the domain-specific models according to any new requirements for the system. After that, the transformation rules will be automatically generated and the legacy source code will be transformed as a consequence. The whole process is transparent and auto-driven by the model interpreters.

The intrinsic benefit of this approach is large-scale adaptation across multiple source files that are driven by model properties. Such adaptation can be accomplished through minimal changes to the models. C-SAW also provides an ability to perform aspect weaving at the modeling level, in addition to the implementation level. The two-level weaving approach will be a unique contribution of this demonstration.

3. ASPECT TRANSFORMATIONS

As argued in [2], it is likely that aspect-oriented programming (AOP) [5] will offer benefits to legacy systems that are coded in languages other than Java. Yet, bringing AOP support to legacy languages requires the availability of a parser for each language, as well as an ability to perform transformations on syntax trees. The demonstration will show our initial work at using DMS as the underlying engine to support aspect weaving.

Several concerns will be woven into an Object Pascal application using lower-level DMS transformation rules. The general meaning of the transformations will be explained and the sequence of actions needs to perform aspect weaving will be demonstrated on the case study. The need for higher-level representations for aspect languages, which translate down to the rules required by DMS, will be motivated and an initial solution will be described. Figure 2 shows our initial ideas for using DMS to generate aspect weavers for various languages. This latter idea is still under investigation and much work is sill needed, but a prototype of the idea will be shown.

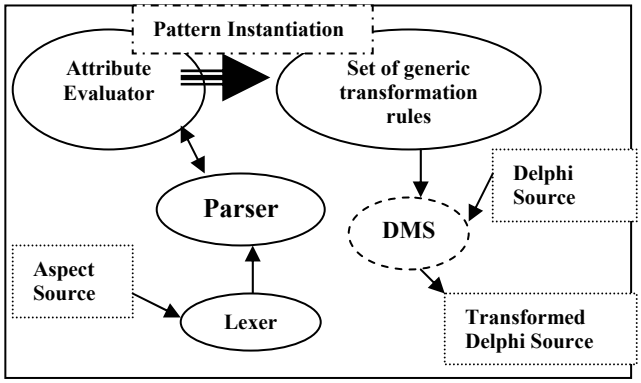

Figure 2. Aspect Weaving with DMS

4. FURTHER INFORMATION

Additional information about C-SAW and GenAWeave, including video demonstrations and related papers, can be found at:

http://www.cis.uab.edu/gray/Research/C-SAW/
http://www.cis.uab.edu/gray/Research/GenAWeave/

5. ACKNOWLEDGEMENT

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

6. REFERENCES

[1] Baxter, I., Pidgeon, C., and Mehlich, M., "DMS: Program Transformation for Practical Scalable Software Evolution," *International Conference on Software Engineering (ICSE)*, Edinburgh, Scotland, May 2004, pp. 625-634.

[2] Gray, J., and Roychoudhury, S., "A Technique for Constructing Aspect Weavers using a Program Transformation Engine," *AOSD '04, International Conference on Aspect-Oriented Software Development,* Lancaster, UK, March 2004, pp. 36-45.

[3] Gray, J., Zhang, J., Lin, Y., Roychoudhury, S., Wu, H., Sudarsan, R., Gokhale, A., Neema, S., Shi, F., and Bapty, T., "Model-Driven Program Transformation of a Large Avionics Framework," *Generative Programming and Component Engineering (GPCE 2004)*, Springer-Verlag LNCS, Vancouver, BC, October 2004.

[4] Karsai, G., Maroti, M., Lédeczi, Á., Gray, J., and Sztipanovits, J., "Composition and Cloning in Modeling and Meta-Modeling," *IEEE Transactions on Control System Technology (special issue on Computer Automated Multi-Paradigm Modeling)*, March 2004, pp. 263-278.

[5] Kiczales, H., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W., "Getting Started with AspectJ," *Communications of the ACM*, October 2001, pp. 59-65.

[6] Zhang, J., and Gray, J., "Legacy System Evolution through Model-Driven Program Transformation," *EDOC Workshop on Model-Driven Evolution of Legacy Systems*, Monterey, CA, September 2004.