Support for domain-specific aspect languages

XAspects is the \texttt{xajc} command together with a library of plug-ins

\texttt{xajc *.java}

A plug-in implements this interface in order to be used by the \texttt{xajc} command

\begin{verbatim}
abstract class AspectPlugin {
    ... ...
}
\end{verbatim}

The \texttt{xajc} command is a wrapper around the AspectJ compiler that introduces macro-like features to allow domain-specific aspect languages to be implemented as if they were an extension to the AspectJ language itself.

The phases of compilation

(1) Programmer uses domain-specific aspects embedded in their regular AspectJ code

(2) The \texttt{xajc} command separates the domain-specific aspects from the rest of the code and passes the aspects to their defining plug-ins

(3) Each plug-in exports the new interfaces (fields, methods, classes) it will define.

(4) The program is compiled with these stub interfaces.

(5) The entire program in byte-code form is passed back to each plug-in so they can do whole-program analysis on the byte-codes.

Use macros that generate AspectJ source code

XAspects

Ankit Shah, Macneil Shonle
\texttt{(ankit,mshonle)@ccs.neu.edu}

Prof. Karl Lieberherr
Northeastern University, Boston MA

The XAspects library currently has plug-ins to support the domains of adaptive programming (as provided by Demeterf), thread coordination (as provided by COOL), and RMI (as provided by RIDL).

Plug-ins are also being written to support “Personalities” (a dynamic alternative to Multiple Inheritance) and the Gang of Four Design Patterns.

Achieve separation of concerns for several domains simultaneously

Direct support for programming techniques in the language eases the development effort. For example, Demeter f shows how “new language constructs can make design patterns disappear” [Hannemann, Kiczales].

Many of the problems that can be solved in AspectJ can only be done so by an expert in the language. The plug-in mechanism allows for aspect programming patterns to be encapsulated as custom languages that can check that the pattern is being used properly.

In the past, several domain-specific languages were created to solve problems. While these domain-specific languages were too narrow to be included in the AspectJ programming language they were useful enough to warrant custom tools to support them. Now with XAspects, these languages can exist in a framework where they can be composed with other languages.

\url{http://www.ccs.neu.edu/home/mshonle/xaspects}