Innovative Uses of Programming Constructs Supporting Aspect-Oriented Programming

Miklós Espák

Separation of concerns is one of the most important objectives during software development. Unfortunately, the boundaries of the individual concerns of a software do not overlap with the boundaries of the corresponding program modules in most cases. When a concern cannot be mapped unambiguously to a single program module, it is said to crosscut these modules. Aspect-oriented programming (AOP) provides a technique for separating crosscutting concerns into distinct program modules. AOP is a substantially new approach in software development. Due to its novelty, the most appropriate language constructs to support it are yet to be found.

By this time several AOP systems have been developed. Naturally, the implementation of these systems cannot break away from current (non- AO) languages. Instead, the impact of current mainstream languages is very strong on AOP system implementations. These languages - being the base of an AO extension of framework - crab the creation and implementation of the most appropriate AO language constructs.

In the paper I will show how typical object-oriented (OO) languages hinder introducing more innovative AO language constructs, and I will provide solutions for these restrictions.

The paper focuses mainly on the following issues:

- designating execution points
- context exposure
- argument passing modes

I will show that by "rehabilitating" some constructs well-known in declarative languages and introducing them in the OO environment, a more expressive AO environment can be gained.

I will present a sample implementation of the new concepts mentioned. The implementation has been worked out in the Common Lisp Object System (CLOS), and makes a part of a universal low level framework, which serves as a new foundation for current and future AOP systems.