AM

Aspect mining is a reverse engineering process that aims at finding crosscutting concerns in existing systems.

Aspect mining is an upcoming research direction aimed at finding crosscutting concerns in existing, non-aspect oriented code. Once these concerns have been identified, they can be used for program understanding or refactoring purposes, for example by integrating aspect mining techniques into the software development tool suite. In addition to that, aspect mining research increases our understanding of crosscutting concerns: it forces us to think about under what circumstances a concern should be implemented as an aspect, it helps us find crosscutting concerns that are beyond the canonical ones such as logging and error handling, and it may lead to concerns that are crosscutting, yet not easily modularized with current aspect technology (such as, e.g., ASPECTJ).

**AO**

The second evaluation challenge is that the decision that a concern is crosscutting and amenable to an aspect-oriented implementation is a design choice, which is a trade-off between alternatives. Thus, there is not a yes/no answer to the question whether a concern identified is suitable for an aspect implementation. As a consequence, quantitative data on the number of false negatives (how many crosscutting concerns are missed) or false positives (how many of the concerns we identified are in fact not crosscutting) has a subjective element to it. This means that an evaluation of an aspect mining technique just in terms of, for example percentages of false positives and negatives, or in terms of precision and recall, is an oversimplification.

**Concerns y AM**

We distinguish between a concern’s intent and extent:

—A concern’s intent is defined as the objective of the concern. For example, the intent of a tracing concern is that all relevant input and output parameters of public methods are appropriately traced.

—A concern’s extent is the concrete representation of that concern in the system’s source code. For example, the extent of the tracing concern consists of the collection of all statements actually generating traces for a given method parameter.

In aspect mining, we search for source code elements that belong to the extent of concerns that crosscut the software system’s modularization structure. Such crosscutting concerns are not dedicated to a modularization unit like a single package, class hierarchy, class, method, but are scattered over all these units. As an example, the tracing concern will affect many different methods distributed over different packages or classes. A consequence of this scattering is tangling: modular units cannot deal exclusively with their core concern, but have to take into account the implementation of other concerns that crosscut their modularization as well.

**AO**

Aspect-oriented software development aims at avoiding the maintenance problems caused by scattering and tangling by making use of the new aspect modularization construct. As a simple example, consider an implementation of the tracing concern in ASPECTJ2, as shown in Figure 1. The declare statement at the top of the aspect body ensures that all classes contained in a particular package extend the Traceable interface, using a so-called inter-type declaration. The Traceable interface itself is provided in the subsequent lines, including a default implementation of the interface. In this way, the aspect extends multiple classes, thereby capturing the statically crosscutting nature of tracing. The remainder of the aspect captures the dynamic crosscutting, using a “pointcut” which intercepts all calls to public methods, and “around advice” that emits a string with the signature of the executing method just before and just after its execution. The aspect can be woven into the base code, keeping the latter oblivious to the tracing concern. This helps to reduce the tangling in the base code and provides a non-scattered implementation of the crosscutting concern. Furthermore, a (small) reduction in code size can be achieved if the crosscutting is sufficiently regular (as is the case with the tracing concern: the pointcut expression can quantify over all public methods).

**Este tiene mucho de aspect mining para explicar**