

## Feature Debugging Tool

Fernando Olivero. Caesar Systems.

**KeyWords:** Feature, Debugging, Code Coverage, Method wrappers, SUnit test

Currently, when dealing with bugs the programmers take a somewhat manual and crafty approach. Once the bug is manifested, the pathway to find the conflict producing it --generally a method incorrectly programmed-- is debugging by steps until one has a clue pointing what could be the cause of the problem. The size of the set of methods to review is inversely proportional to the knowledge of the design one has. The feature debugging tool goal is to aid the programmer in finding the problematic method/s, providing an automated way.

The concept of *feature* is introduced in order to accomplish that. A feature is a single unit of functionality, for example a SUnit test. We define the *coverage of a feature* as the set of the methods that are called when executed.

In the case of a test, the tool can find the code coverage by running the suite with an adequate setup of *method wrappers*. It wraps all desired methods prior to the execution of the feature, and after running retrieves only the called ones. The condition for selecting desired methods is a setting of the tool. It could be for example wrapping all methods modified after a specific date or all methods of a given version.

The coverage comparison between selected features is then performed, by sorting the called methods of each feature by number of occurrences in the coverage of all the other selected features. This categorizes them as exclusive or shared properties.

The latter classification is the vehicle for narrowing the set of methods one should take a closer look at, because in case running a feature yields an error, exclusive methods have bigger chance of being incorrectly programmed than ones shared with others.

Debugging the feature and halting on the called methods with higher probability of inducing errors is another handy service that the tool offers.