Unit Testing and Friends in C++

Gábor Márton, Zoltán Porkoláb

Unit testing of object-oriented code requires two fundamental artifacts: replacing some of the original code with some kind of test doubles (aka dependency injection), and providing extra access to these private dependencies. In most cases we cannot apply link-time or preprocessor based dependency injection, therefore a typical testable unit uses additional interfaces or template type parameters for pure testing purposes. Thus a unit which we developed without testing in mind might have to be changed intrusively for testing. These changes can take us far away from the original natural design of the unit. As a result, in C++, test code is often badly interwoven with the unit we want to test. We've seen such interleaving in the software world before: exceptions and aspect oriented programming were both invented to try to eliminate this kind of interleaving of independent aspects.

In this paper we demonstrate the above problems using code examples to show the decay of original intentions. We discuss current possible directions to solve the dependency injection problem: e.g. using compile-time seams [1]. Then, we overview the current (partial) solutions for adding extra access for private data, like using friends, the Attorney-Client idiom [2], or exploiting privates using explicit template instantiations.

Regarding to extra access of private data, we present how a minimal language extension would help and make the testing code less intrusive. Open non-intrusive friend declarations could provide a good method to separate the test related code from the original intentions. We implemented a proof-of-concept solution, based on LLVM/Clang to show that such constructs can be established with a minimal syntactical and compilation overhead.

References

[1] M. Rüegg, *Refactoring Towards Seams in C++*, ACCU OVERLOAD, #108, ISSN 1354-3172, April 2012.

[2] A. R. Bolton, *Friendship and the Attorney-Client Idiom* Dr.Dobbs's Journal, January 01, 2006.