Automatic Test Generation for Space

Ulisses Araújo Costa

Department of Informatics, University of Minho
Campus de Gualtar, 4710-057 Braga, Portugal
ulissesmonhecosta@gmail.com

Abstract. ESA (European Space Agency) uses a big engine to perform tests in the Ground Segment infrastructure, specially the Operational Simulator. This engine uses many different tools to ensure the development of regression testing infrastructure and these tests perform black-box testing to the C++ simulator implementation. VST (Vision Space Technologies) is one of the companies that provides these services to ESA and they need a tool to instead of writing manually scripts to perform tests, it should try to automatically infer tests from the existing C++ code. There are many approaches trying to tackle this problem and therefore many tools were developed. Here we will present a study on the most recent tools that uses: Model-based testing, Specification-based testing, Constraint-based generation, Random generation and Grammar-based generation for the most used languages - C, JAVA and C#.
Automated Test Case Generation tools give support for creating test cases and at the same time ensure test case coverage methodically. The main goal of this tools is extract information from the program on how to generate executable test cases. Using manual written tests is tedious, time consuming and error-prone. Lots of functions/methods need full code coverage and this technique leaves to incomplete test suites and is hard to create tests that cover specific code paths potentially leaving many hidden bugs. Besides that, software is not a static artifact and is constantly evolving, so a test generation technique could be a more suitable mechanism in the development process. We will be mainly focused on the white-box automatic test generation techniques.
Since no other testing tool used by ESA makes any use of a formal model we decide to start by infer the UML model from the existing C++ code and try to explore the generation of Object Constraint Language specifications from C++ code and UML diagrams. The idea is to capture the requirements properties about the code, and from this beef up the white-box automatic test generation.