High-level Restructuring of TTCN-3 Test Suites

Antal Wu-Hen-Chang, Dung Le Viet, and Gyula Csopaki

Telecommunication software provides the foundation of the communication infrastructure. These systems must be reliable, efficient and compatible with systems from different vendors. Consequently, their development must be accompanied by quality assurance activities, thus testing plays a vital role during the development process of each telecommunication system. The purpose of testing is to find all errors and shortcomings of the system. This is a very resource-demanding and time-consuming task, because it requires the manual effort of many well-trained developers, therefore its support is an important challenge.

TTCN-3 (Testing and Test Control Notation 3) [1] is the new industry-standard test specification language that was developed and standardized by the European Telecommunication Standards Institute (ETSI). It is a very powerful language that has been tried out on different application areas of testing. It can be applied for all kinds of black-box testing for reactive and distributed systems and it is suitable to test virtually any system including telecommunication and mobile systems, Internet and CORBA based protocols.

The general testing process with TTCN-3 includes the following main steps: The developed test suite is compiled and extended with an adaptor that provides the connection between the tested system and the executable test suite. Then the executable test suite is executed against the system under test. Finally, the results are evaluated.

TTCN-3 has a special language element – the template – that provides sophisticated means for describing test data. In order to test complicated systems, the TTCN-3 templates can be created either in a manual or in an automatic way, but in neither case is the result optimal, since developers cannot cope with the enormous number of huge data structures, and automatic methods focus primarily on the generation problem. According to our empirical experiences test data definition occupies at least 60-70 percent of a complete test specification and they are highly redundant. Consequently, these modules are unnecessarily large that leads to several problems. In case of very large TTCN-3 modules the compilation time can be surprisingly long, which sets back the development process. It is not uncommon, that the compilation of the test specification takes more than an hour on an average computer and complicated test suites consist of several different modules. Besides, executable test suites derived from large modules have performance drawbacks, that makes it harder to develop performance or scalability test suites, where performance is a critical issue. Furthermore, we must take into account, that the development process of a test suite is cyclic, therefore these problems appear repeatedly.

By eliminating the redundant and unused data structures, the quality of the generated implementation code can be significantly improved. In our paper, we introduce a re-engineering method that can be applied without human intervention to test data templates defined in TTCN-3. The approach analyzes and restructures [2, 3] an already existing TTCN-3 template specification, so that it becomes more compact, redundancy-free and the compilation time is reduced. Naturally, the alterations retain semantical correctness, only syntactical changes are introduced.

Beside the static test data restructuring, it is worth analysing the TTCN-3 description of the dynamic behavior. The test specifications often contain repeated test steps, but because of their complex nature, they can't be detected easily. We examine the possibility of applying dynamic behavior restructurings to TTCN-3, that would extract and reuse frequently appearing test steps.

References

[1] ETSI ES 201 873-1 V2.2.1 (2002-08). Methods for Testing and Specification (MTS); The Testing and Test Control Notation Version 3; Part1: TTCN-3 Core Language.

[2] E.J. Chikofsky and J.H. Cross. Reverse Engineering and Design Recovery: A Taxonomy, *IEEE Software*, Vol 7(1):13-17, 1990.

[3] T. Mens and T. Tourwe. Survey of Software Refactoring, *IEEE Trans. on Software Engineering*, Vol. 30(2):126-139, 2004.