Automation of Refactoring and Refactoring Suggestions for TTCN-3 Test Suites
The TRex TTCN-3 Refactoring and Metrics Tool

Helmut Neukirchen and Benjamin Zeiss
Software Engineering for Distributed Systems Group,
Institute for Informatics, University of Göttingen,
Lotzestr. 16–18, 37083 Göttingen, Germany
{neukirchen|zeiss}@cs.uni-goettingen.de

Abstract *Refactoring is not only useful for source code of implementations, but as well for test specifications. The open source TRex tool automates the application of refactorings and the detection of refactoring opportunities for test suites that are specified using the standardised* Testing and Test Control Notation *(TTCN-3). Depending on the refactoring, the behaviour preserving transformations may include syntax tree transformations and direct modification of the source code; for suggesting refactorings, metrics are calculated and code smell patterns are detected.*

Introduction. The *Testing and Test Control Notation* (TTCN-3) [1] is a mature standard from the telecommunication and data communication domain that is widely used in industry and standardisation to specify and execute test suites. Just like any other software artifact, tests suffer from quality problems [2]. To remove such quality problems from TTCN-3 test suites, we use refactoring [3]. For suggesting refactorings, we use a combination of metrics and code smell detection.

In the following, we first present our approach for the quality assessment and improvement of TTCN-3 test suites. Subsequently, the TTCN-3 Refactoring and Metrics Tool TRex and its implementation are described. Finally, future work is discussed in the outlook.

Refactoring, Metrics, and Code Smell Detection for TTCN-3 Test Suites. Refactoring of test suites has so far only been studied in the context of JUnit [2]. Thus, we have developed a refactoring catalogue for TTCN-3 [4, 5] which includes 23 refactorings using language specific concepts of TTCN-3. Furthermore, we found 28 refactorings from Fowler's Java refactoring catalogue [3] to be applicable to TTCN-3.

For being able to automatically identify locations in source code where a refactoring is worthwhile, we investigated corresponding TTCN-3 metrics and TTCN-3 code smells. For example, a *Number of References* metric is used to identify definitions that are never referenced and can thus be removed or that are referenced only once and can thus be inlined using a corresponding refactoring. Even though we experienced that metrics are able to detect various issues, they are not sophisticated enough to detect more advanced problems. Therefore, we investigated pattern-analysis of source code and as a result, we have developed a catalogue of TTCN-3 code smells [6]. So far 38 TTCN-3 code smells have been identified.

TRex Implementation. To automate refactoring for TTCN-3 test specifications, we have implemented the open source TTCN-3 Refactoring and Metrics tool TRex [4]. The initial version has been developed in collaboration with Motorola Labs, UK [7]. TRex implements state-of-the-art editing capabilities, assessment and improvement techniques for TTCN-3 test suites based on the calculation of metrics, automated smell detection, and refactoring. TRex is based on the Eclipse Platform [8] and thus makes use of the infrastructure offered, e.g. the *Language Toolkit* (LTK) or the *Eclipse Test & Performance Tools Platform* (TPTP) static analysis framework. The analysis infrastructure including lexer, parser, symbol table, pretty printer, etc. for TTCN-3 have been implemented using *ANother Tool for Language Recognition* (ANTLR) [9].

The automated refactorings we currently provide concentrate mostly on the improvement of test data descriptions (TTCN-3 templates). The refactoring implementations can be applied in two different ways: either the test developer invokes the refactoring from the code location

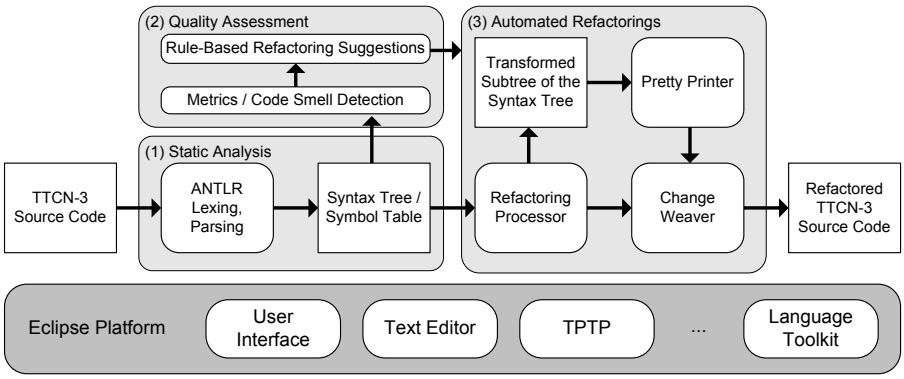

Figure 1: The TRex Toolchain

which should be subject to the refactoring or the refactoring is invoked directly by a quick-fix which is provided by the analysis results of the automated quality assessment.

For the assessment, a considerable number of size metrics (such as counting the number of references to a definition) and structural metrics (using control-flow graphs and call graphs) are calculated. Furthermore, a total number of 11 TTCN-3 code smell detection rules have been implemented that partially allow the use of quick-fixes to invoke automated refactorings.

The overall toolchain of TRex is depicted in Fig. 1. Based on the syntax tree and symbol table, the automated refactorings can either be applied directly or invoked through the refactoring suggestions obtained by means of metrics and code smell detection. The refactorings are applied directly to the source code using a programmatic text editor and may as well involve syntax tree transformations. The corresponding text representation from transformed subtrees is reobtained by a pretty printer component and weaved back into the surrounding TTCN-3 source code.

The implementations of metric calculation, code smell detection, and refactoring are tested using *Plug-in Development Environment* (PDE) JUnit tests, e.g. by comparing source code snippets before and after the refactoring.

Outlook. A remaining issue open for research is the validation of test refactorings: Unlike Java refactorings, for example, there are no unit tests available for tests and simply running a test suite against an implementation is not enough if the test behaviour consists of more than one path. We are thus investigating bisimulation to validate that the observable behaviour of a refactored test suite has not changed. In addition, we are extending TRex by implementing further TTCN-3 refactorings and more sophisticated code smell detection techniques.

References

[1] ETSI: ETSI Standard (ES) 201 873 V3.2.1: The Testing and Test Control Notation version 3; Parts 1-8. European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France (2007)

[2] van Deursen, A., Moonen, L., van den Bergh, A., Kok, G.: Refactoring Test Code. In: XP2001. (2001)

[3] Fowler, M.: Refactoring – Improving the Design of Existing Code. Addison-Wesley, Boston (1999)

[4] TRex Team: TRex Website. `http://www.trex.informatik.uni-goettingen.de` (2007)

[5] Zeiss, B.: A Refactoring Tool for TTCN-3. Master's thesis, Institute for Informatics, University of Göttingen, Germany, ZFI-BM-2006-05 (2006)

[6] Bisanz, M.: Pattern-based Smell Detection in TTCN-3 Test Suites. Master's thesis, Institute for Informatics, University of Göttingen, Germany, ZFI-BM-2006-44 (2006)

[7] Baker, P., Evans, D., Grabowski, J., Neukirchen, H., Zeiss, B.: TRex – The Refactoring and Metrics Tool for TTCN-3 Test Specifications. In: TAIC PART 2006, IEEE Computer Society (2006) 90–94

[8] Eclipse Foundation: Eclipse. `http://www.eclipse.org` (2007)

[9] Parr, T.: ANTLR parser generator v2. `http://www.antlr2.org` (2007)