Quality Assurance for TTCN-3 Test Specifications

Helmut Neukirchen

Software Engineering for Distributed Systems Group,
Institute for Computer Science,
University of Göttingen, Germany

Outline

1. Introduction
2. TTCN-3 Metrics and Code Smells
3. TTCN-3 Refactoring
4. TRex Tool
5. Summary / Outlook

1. Introduction: TTCN-3

- Testing and Test Control Notation version 3
  - Language for specifying and implementing distributed tests.
  - Standardised by European Telecommunications Standards Institute (ETSI).
- High abstraction level increases productivity and reusability:
  - Abstract from low-level communication and data encoding/decoding.
    ⇒Test independent from lower layers of System under Test (SUT).
  - Well defined interface to adaptation layer.
    - Implements communication mechanisms and encoding/decoding.

    ⇒Allows to execute abstract TTCN-3 test specifications.

TTCN-3: Past and Present

- Origin in telecommunication domain:
  - Standardisation bodies publish (e.g. for ISDN, GSM, UMTS):
    - Specification of a communication protocol,
    - Test suite to test conformance of an implementation to its specification.
  - Industry:
    - Implements specified protocols in their equipment,
    - Implements test adaptation layer and executes standardised test suites against their implementation.
- Today:
  - TTCN-3 not only used in telecommunication standardisation domain, but also in domains like Internet, Service-Oriented Architectures, Automotive, …

TTCN-3 Example

- Look and feel of common programming language:

```
module exampleModule {
   ...
   type record IpAddressType { charstring ipAddress };
   template IpAddressType localhostTemplate := {
      ipAddress := "127.0.0.1"
   }
   testcase exampleTestCase() runs on ExampleComponent {
      portA.send(localhostTemplate);
      alt {
         [] portB.receive(localhostTemplate) {
               setverdict(pass);
            }
         [] portB.receive(IpAddressType:{*}) {
               setverdict(fail);
            }
      }
   }
}
```


Motivation

- Huge TTCN-3 test suites (>40000 LOC), e.g. ETSI test suites for:
  - Session Initiation Protocol (SIP),
  - Internet Protocol Version 6 (IPv6).
  - Suffer from quality problems like any larger software!

⇒Quality assessment and improvement required!

External Quality of Test Specifications

- ISO 9126 software product quality model:
  - External quality:
    quality of test in relation to external environment.
    - E.g. correctness of tests,
      coverage obtained by executing tests against SUT.
    - However, SUT often not available during test development:
      - Standardisation: Only abstract test suites are developed.
        (not executable because adaptation layer is missing).
      - Industry: SUT and tests are developed in parallel and thus SUT is not always available.

Internal Quality of Test Specifications

⇒Consider internal quality:

- Quality of test specification on its own.
  - Usually determined by static analysis, e.g.
    compiler warnings, metrics, code anomaly analysis.
- Allows to assure test quality during development!
- Allows to assess properties of source code,
  e.g. maintainability.

Example Quality Deficiencies

- Excerpt from standardized SIP TTCN-3 test suite:

```
function ptc_CC_PR_TR_CL_TI_015(CSeq loc_CSeq_s )
         runs on SipComponent
{
    var Request v_BYE_Request;

    initPTC(loc_CSeq_s);
    v_Default := activate(defaultCCPRPTC());

    tryingPTCBYE();

    waitForTimeout(65.0*PX_T1);

    notRepeatBYE(PX_TACK);

} //end ptc_CC_PR_TR_CL_TI_015
```

Variable is never used!

Default for alternatives activated, but never deactivated.

Hard coded "magic" values.

Quality Assessment and Improvement for TTCN-3 Test Suites

- Approach:
  - Assess test suites,
  - Detect issues,

  → **Metrics, Smell Detection**
  - Restructure test suites. → **Refactoring**

Outline

1. Introduction
2. **TTCN-3 Metrics and Code Smells**
3. TTCN-3 Refactoring
4. TRex Tool
5. Summary / Outlook

2. TTCN-3 Metrics (Example)

- Developed using the Goal, Question, Metrics approach.
  - Basili, Weiss: A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on SE, 1984
- Goal: Improve readability of TTCN-3 source code.
- Question: "Are any definitions unused or used only once?"
  - Count number of references to definitions.

⇒Metric: Number of References to Definitions

Neukirchen, Zeiss, Grabowski, Evans, Baker:
Quality assurance for TTCN-3 test specifications.
Software Testing, Verification and Reliability, 2008.

Bad Smells in TTCN-3 Test Suites

- Metrics sometimes not powerful enough, e.g.:
  - Goal: Improve changeability of TTCN-3 source code.
  - Question:
    "Do local changes require further non-local changes?"
    - Find duplicated code.

  ⇒ Pattern-based approach required: code smells.
  - Patterns of inappropriate usage of TTCN-3.
  - By definition not a smell:
    - Syntax errors,
    - Violation of static semantics,
    - Defects in test case logic.

TTCN-3 Code Smells

- Collected TTCN-3 code smells in a structured catalogue.
- So far identified 38 TTCN-3 code smells with respect to:
  - Duplicated Code, References, Parameters, Complexity, Default Anomalies, Test Behaviour, Test Configuration, Coding Standards, Data Flow Anomalies, Miscellaneous.

  Neukirchen, Zeiss, Grabowski:
  An Approach to Quality Engineering of TTCN-3 Test Specifications.
  International Journal on Software Tools for Technology Transfer, 2008.
- Smells only give hints:
  - What is considered as smell, may vary from project to project.

TTCN-3 Code Smell (Example): Activation Asymmetry

- **Description:**
  - Default activation and deactivation are not inside the same statement block.
- **Motivation:**
  - Improve analysability with respect to active defaults.
  - Enable static analysis of matching default activation and deactivation.
- **Options:**
  - A missing deactivate may not be considered as code smell inside TTCN-3 testcase constructs, since defaults are implicitly deactivated at the end of a testcase.
- **Related Action(s):**
  - Add default deactivation (or activation) if missing.
  - Move matching default activation and deactivation into same statement block.

TTCN-3 Code Smell (Example): Activation Asymmetry

- **TTCN-3 Example:**

```
module ExampleModule {
  function exampleFunction() return default {
    return activate(exampleAltstep());
  }
  testcase exampleTestcase() runs on ExampleComponent {
    var default myDefaultVar := null;
    myDefaultVar := exampleFunction();
    alt {
      [] portA.receive(messageOne) { portB.send(messageTwo); }
    }
    deactivate(myDefaultVar);
  }
}
```


Outline

1. Introduction
2. TTCN-3 Metrics and Code Smells
3. **TTCN-3 Refactoring**
4. TRex Tool
5. Summary / Outlook

3. Refactoring

„A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior."

Fowler: Refactoring – Improving the Design of Existing Code. Addison-Wesley, 1999

- A refactoring can be used to remove a code smell.
- TTCN-3 refactoring catalogue:
  - More than 50 refactorings for improving TTCN-3 test suites:
    - Test behaviour, test data, overall test suite structure.

    Neukirchen, Zeiss, Grabowski, Evans, Baker:
    Quality assurance for TTCN-3 test specifications.
    Software Testing, Verification and Reliability, 2008.

Example: Inline Template Parameter

- **Summary:**
  - Inline a template parameter which is always given the same actual value.
- **Motivation:**
  - Unneeded parameters create
    - code clutter,
    - more coupling than needed.
- **Mechanics:**
  - Copy template.
    - Remove parameter from formal parameter list,
    - Replace each reference to formal parameter inside the template with the common actual parameter value.
  - Remove actual parameter from each template reference.
  - Remove original template.

Example: Inline Template Parameter

- **TTCN-3 example (unrefactored):**

```
module ExampleModule {

  template ExampleType exampleTemplate(charstring addressParameter):={
    ipv6:=false,
    ipAddress:=addressParameter
  }

  testcase exampleTestCase() runs on ExampleComponent {
    portA.send(exampleTemplate("127.0.0.1"));
    portB.receive(exampleTemplate("127.0.0.1"));
  }

}
```


Example: Inline Template Parameter

- **TTCN-3 example (refactored):**

```
module ExampleModule {

  template ExampleType exampleTemplate:={
    ipv6:=false,
    ipAddress:="127.0.0.1"
  }

  testcase exampleTestCase() runs on ExampleComponent {
    portA.send(exampleTemplate);
    portB.receive(exampleTemplate);
  }

}
```


Rule-Based Quality Assessment & Improvement

- Metrics-based:
  - Number of references to a template = 0
    ⇒ *Remove Template*.
  - Number of references to a template = 1
    ⇒ *Inline Template*.
- Code Smell-based:
  - Identical actual parameter value
    ⇒ *Inline Template Parameter*.
  - Duplicate branches in alt statements
    ⇒ *Extract Altstep*.

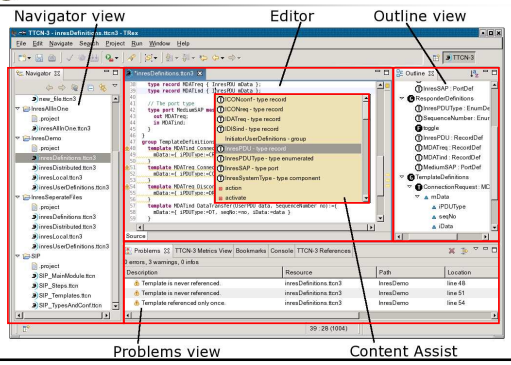
Outline

1. Introduction
2. TTCN-3 Metrics and Code Smells
3. TTCN-3 Refactoring
4. **TRex Tool**
5. Summary / Outlook

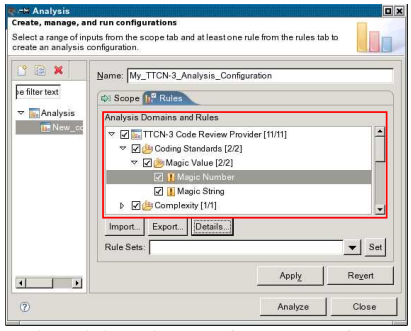
4. TRex

- TTCN-3 Refactoring and Metrics tool:
  - Open source plug-ins for Eclipse platform,
  - Integrated TTCN-3 development environment (advanced editing),
  - External compiler integration,
  - Automated calculation of TTCN-3 metrics,
  - Automated detection of TTCN-3 code smells using static analysis,
  - Tool supported TTCN-3 refactoring,
  - Rule-based quality assessment & improvement.
    - Refactorings associated to code smells as a "Quick Fix".

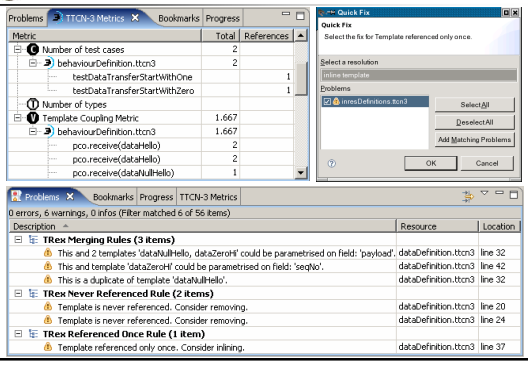
Editing



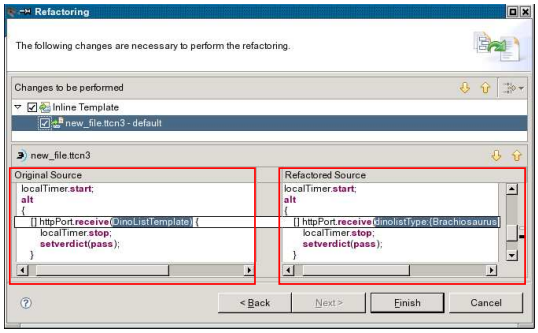
TTCN-3 Code Smell Detection Configuration



Rule-Based Issue Detection



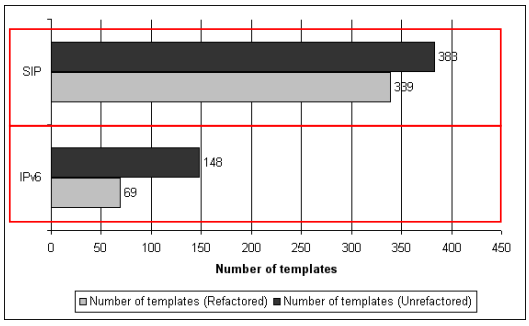
TTCN-3 Code Smell Removal



Code Smells detected in ETSI Test Suites

Metric / TTCN-3 Code Smell	SIP	IPv6
Lines of code	42397	46163
Number of testcases	528	295
Duplicate Alt Branches	938	224
Activation Asymmetries	73	317
Distinct Magic Values	135	222
Unused Definitions	50	156

Automated Reduction of Templates in Test Suites



Outline

1. Introduction
2. TTCN-3 Metrics and Code Smells
3. TTCN-3 Refactoring
4. TRex Tool
5. **Summary / Outlook**

5. Summary and Outlook

- Summary:
  - Quality assessment for TTCN-3 test suites using metrics and smells.
  - Quality improvement of TTCN-3 test suites using refactoring.
  - TRex tool for automated quality assurance of TTCN-3 test suites.
  - Results from standardised test suites.
- Outlook:
  - Continue development of open-source TRex tool.
  - Simulation of TTCN-3 test suites to assess dynamic properties.

- Thank you for your attention!
- Any Questions?
- TRex and further publications available from

http://www.trex.informatik.uni-goettingen.de