

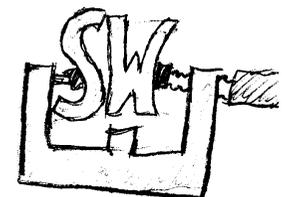
# Pattern-Based Quality Assessment of TTCN-3 Test Suites

---

Helmut Neukirchen



Software Engineering for Distributed Systems Group  
Institute for Informatics  
Georg-August-University Göttingen  
Germany



<http://www.swe.informatik.uni-goettingen.de>

# Outline

---

1. Introduction
2. TTCN-3 Code Smells
3. TRex Tool
4. Summary / Outlook

# 1. Introduction

---

- Testing and Test Control Notation version 3:
  - Language for specifying and implementing distributed tests.
  - Standardised by European Telecommunications Standards Institute (ETSI).
- Huge TTCN-3 test suites (>40000 LOC), e.g. for:
  - Session Initiation Protocol (SIP),
  - Internet Protocol Version 6 (IPv6).
- Suffer from quality problems like any larger software!

# Motivation

- Excerpt from standardised SIP 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 required!

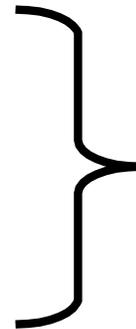
# Quality Assurance for TTCN-3 Test Suites

---

- Initial approach:

- Assess test suites,

- Detect issues,



→ **Metrics**

- Restructure test suites.

→ **Refactoring**

# TTCN-3 Metrics (Example)

---

- 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

Zeiss, Neukirchen, Grabowski, Evans, Baker:  
Refactoring and Metrics for TTCN-3 Test Suites.  
SAM'06, LNCS 4320, Springer, 2006.

Zeiss, Neukirchen, Grabowski, Evans, Baker:  
TRex – An Open-Source Tool for Quality Assurance of TTCN-3 Test Suites.  
CONQUEST '06, dpunkt, 2006.

# 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

- TTCN-3 refactoring catalogue:
  - More than 50 refactorings for improving TTCN-3 test suites:
    - Test behaviour, test data description, overall test suite structure.

# TTCN-3 Refactoring (Example): Inline Template

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

```
module ExampleModule {  
  
    template ExampleType exampleTemplate:={  
        ipv6:=false,  
        ipAddress:="127.0.0.1"  
    }  
  
    testcase exampleTestCase() runs on ExampleComponent {  
        portA.send(exampleTemplate);  
    }  
  
}
```

# TTCN-3 Refactoring (Example): Inline Template

---

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

```
module ExampleModule {
```



```
    testcase exampleTestCase() runs on ExampleComponent {  
        portA.send(ExampleType:{false, "127.0.0.1"});  
    }  
}
```

Zeiss, Neukirchen, Grabowski, Evans, Baker:  
Refactoring and Metrics for TTCN-3 Test Suites.  
SAM'06, LNCS 4320, Springer, 2006.

# Rule-Based Quality Assessment & Improvement

---

- Example:
  - Number of references to a template = 0  
⇒ Remove template.
  - Number of references to a template = 1  
⇒ Inline template.

# Outline

---

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

## 2. Code Smells

---

- 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**.

- **“certain structures in the code that suggest (sometimes they scream for) the possibility of refactoring”**

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

# TTCN-3 Code Smells

---

- TTCN-3 code smells:  
patterns of inappropriate usage of TTCN-3.
  - Not considered as TTCN-3 code smell:
    - Syntax errors,
    - Violation of static semantics,
    - Defects in test case logic.
- Notion of metrics and code smells not disjoint:
  - Code smell → Metric: count occurrences of code smell.
  - Metric → Code smell: metric violates boundary.

# TTCN-3 Code Smell Catalogue

---

- Collected TTCN-3 code smells in a structured catalogue.
- So far identified 38 TTCN-3 code smells with respect to
  - Duplicated Code, e.g. Duplicate Alt Branches
  - References, e.g. Singular Component Variable/Const./Timer
  - Parameters, e.g. Constant Actual Parameter Value
  - Complexity, e.g. Complex Conditional
  - Default Anomalies, e.g. Activation Asymmetry
  - Test Behaviour, e.g. Missing Verdict
  - Test Configuration, e.g. Idle Parallel Test Component
  - Coding Standards, e.g. Magic Values
  - Data Flow Anomalies, e.g. Unused Variable Definition
  - Miscellaneous, e.g. Over-specific Runs On

# TTCN-3 Code Smell Description

---

- Fixed format:
  - Name,
  - Description,
  - Motivation,
  - Options,
  - Related Action(s),
  - Example.

# TTCN-3 Code Smell (Example): Activation Asymmetry

---

- **Description:**
  - Default activation and deactivation are in different statement blocks.
- **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);  
    }
```

```
}
```

# Utilising TTCN-3 Code Smells for Advanced Test Suite Quality Assurance

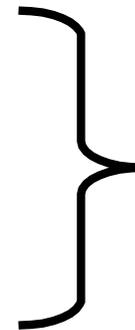
---

- Advanced approach:

- Assess test suites,

- Detect issues,

- Restructure test suites.



**Metrics,  
TTCN-3  
Code Smells**



**Refactoring**

# Outline

---

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



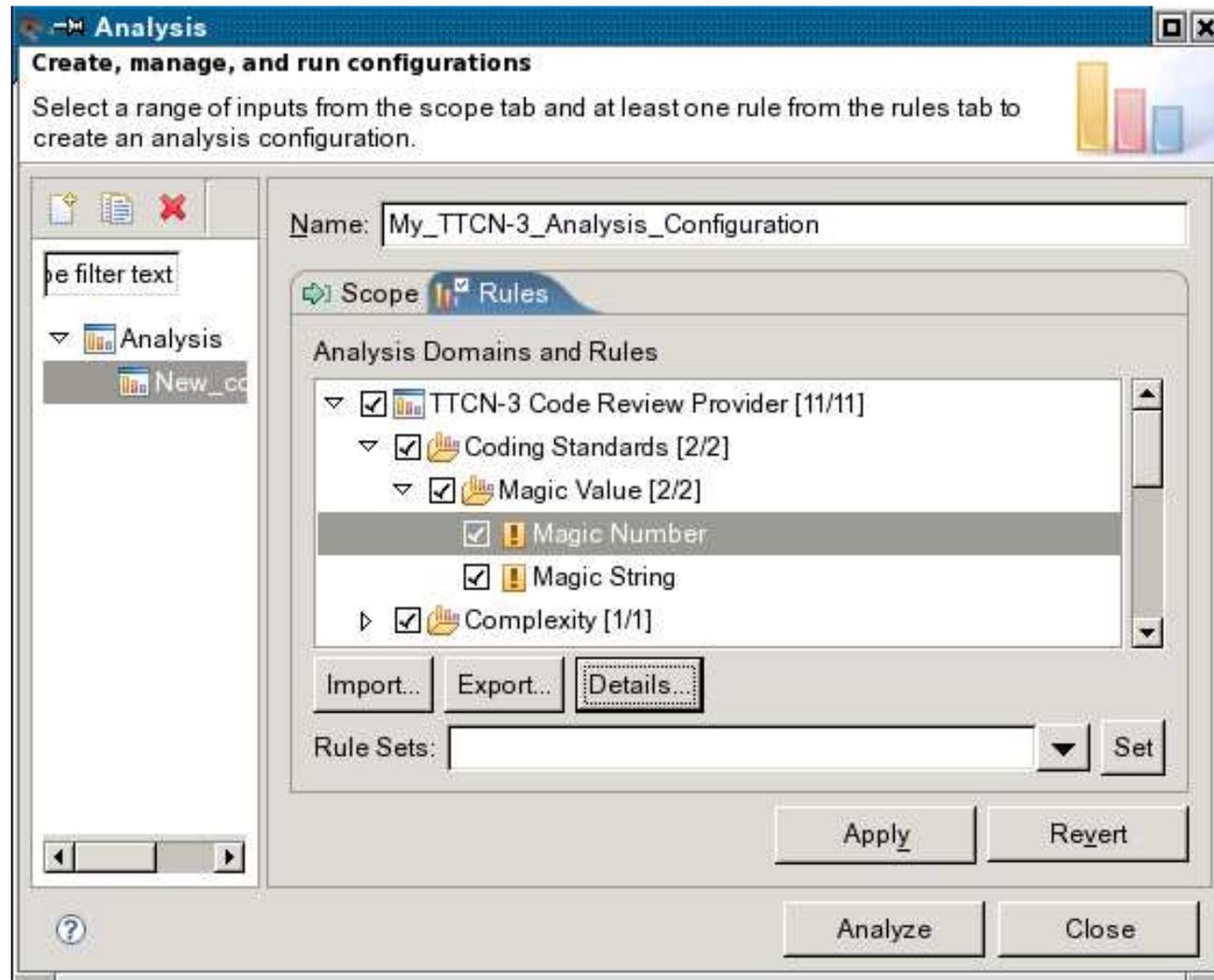
## 3. TRex

---

- TTCN-3 Refactoring and Metrics tool:
  - Open source plug-ins for Eclipse platform,
  - Integrated TTCN-3 development environment,
  - Automated calculation of TTCN-3 metrics,
  - Automated detection of TTCN-3 code smells,
    - Based on Eclipse Test & Performance Tools Platform (TPTP).
  - Tool supported TTCN-3 refactoring,
  - Rule-based quality assessment & improvement.
    - Refactorings associated to code smells as a “Quick Fix”.



# TTCN-3 Code Smell Detection Configuration





# TTCN-3 Code Smell Detection Results

A screenshot of an IDE's 'Analysis Results' window. The window has a blue header with tabs for 'Problems', 'Analysis Results', 'Bookmarks', and 'TTCN-3 Metrics'. The 'Analysis Results' tab is active. The main area shows a tree view of analysis results. The root node is 'ttn-3 all [TTCN-3 Code Review Provider] (30. Januar 2007)'. It has a sub-node 'TTCN-3 Code Review Provider [10 results in 246ms]', which in turn has a sub-node 'Coding Standards [10 results in 49ms]'. Under 'Coding Standards' is a sub-node 'Magic Value [10 results in 49ms]', which is highlighted in blue. Under 'Magic Value' is a sub-node 'Magic Number [10 results in 47ms]'. This node contains ten individual error items, each with a yellow warning icon and a text description: 'inresDistributed.ttcn3:12 Magic Number', 'inresDistributed.ttcn3:13 Magic Number', 'inresDistributed.ttcn3:14 Magic Number', 'inresDistributed.ttcn3:87 Magic Number', 'inresDistributed.ttcn3:93 Magic Number', 'inresLocal.ttcn3:10 Magic Number', 'inresLocal.ttcn3:33 Magic Number', 'inresLocal.ttcn3:39 Magic Number', and 'inresLocal.ttcn3:8 Magic Number'.



# TTCN-3 Code Smell Removal

A screenshot of the "Refactoring" dialog box in an IDE. The dialog has a title bar with a close button. Below the title bar, it says "The following changes are necessary to perform the refactoring." There is a section titled "Changes to be performed" with a list of items: "Inline Template" and "new\_file.tcn3 - default", both with checkboxes and icons. Below this, the file "new\_file.tcn3" is selected. The main area is split into two panes: "Original Source" and "Refactored Source". Both panes show code snippets. The "Original Source" pane shows a code block with a function call `httpPort.receive(DinoListTemplate)`. The "Refactored Source" pane shows the same code block but with the function call `httpPort.receive(dinolistType:{Brachiosaurus`. At the bottom of the dialog are buttons for "< Back", "Next >", "Finish", and "Cancel".

The following changes are necessary to perform the refactoring.

Changes to be performed

- Inline Template
- new\_file.tcn3 - default

new\_file.tcn3

Original Source	Refactored Source
<pre>localTimer.start; alt {   [] httpPort.receive(DinoListTemplate) {     localTimer.stop;     setverdict(pass);   } }</pre>	<pre>localTimer.start; alt {   [] httpPort.receive(dinolistType:{Brachiosaurus     localTimer.stop;     setverdict(pass);   } }</pre>

< Back   Next >   Finish   Cancel



# Application of TRex

Metric / TTCN-3 Code Smell	SIP	IPv6
Lines of code	42397	46163
Number of testcases	528	295
Number of functions	785	643
Number of altsteps	10	11
Number of components	2	10
Duplicate Alt Branches (inside same module only)	938	224
Activation Asymmetries (testcase included)	602	801
Activation Asymmetries (testcase excluded)	73	317
Magic Values (Magic numbers only, 0 and 1 excluded)	543	368
Unused Definitions (local definitions only)	50	156
Singular Component Variables/Constants/Timers	2	15

# Outline

---

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

# 4. Summary and Outlook

---

- Summary:

- Quality assurance for TTCN-3 test suites based on metrics, pattern-based code smells, refactoring for TTCN-3.
- Catalogue of 38 TTCN-3 code smells.
- TRex tool for automated quality assurance of TTCN-3 test suites.

- Outlook:

- Make TTCN-3 code smell catalogue available as Wiki.
- Instead of imperative implementation of TTCN-3 code smell pattern detection:  
Declarative description of TTCN-3 code smell patterns.

- 
- Thank you for your attention!
  - Any Questions?



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