# Quality Assurance of TTCN-3 Test Suites

Dr. Helmut Neukirchen

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

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

# Outline

1. Introduction
2. Metrics / Smells
3. Refactoring
4. TRex Tool
5. Related Work
6. Summary / Outlook

# 1. Introduction: TTCN-3

- Testing and Test Control Notation version 3
  - Language for specifying distributed tests.
  - Standardised by
    - European Telecommunications Standards Institute (ETSI),
    - International Telecommunication Union (ITU).
- History:
  - Standardisation bodies publish (e.g. for ISDN, GSM, UMTS):
    - Specification of a communication protocol,
    - Test suite to check conformance of a protocol implementation to its specification.
  - Industry:
    - Implements specified protocols in their equipment,
    - Execute standardised test suites against their implementation.
- Today:
  - TTCN-3 not only used in telecommunication domain, but for Internet, Service-Oriented Architectures, Automotive, …

# Introduction: TTCN-3

- Example:

```
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);
            }
      }
   }
}
```

- Look and feel of common programming languages:
  - Quality problems like any other source code.

# Motivation

- Huge legacy test suites at Motorola:
    - Migration to TTCN-3.
    - Automatic conversion of a UMTS test suite:
        - 60,000 lines of TTCN-3,
        - Hard to read, use, re-use, maintain.

- Current TTCN-3 tools:
    - Editing, Compiling, Test execution.
    - But: **No support for improving and assessing test suites!**

# Approach

- Assess test suites,
- Detect issues,

→ **Metrics, Smell Detection**

- Restructure test suites.

→ **Refactoring**

# Outline

1. Introduction
2. **Metrics / Smells**
3. Refactoring
4. TRex Tool
5. Related Work
6. Summary / Outlook

# 2. Metrics

**“You cannot control what you cannot measure.”**

De Marco: *Controlling Software Projects*.
Yourdon Press, 1982

⇒ TTCN-3 metrics

- Developed using the Goal, Question, Metrics approach.
  - Basili, Weiss: *A Methodology for Collecting Valid Software Engineering Data*. IEEE Transactions on SE, 1984

# Example
# TTCN-3 Metrics

- Goal: Improve readability of TTCN-3 source code.
  - Question: "Are there any unused definitions?"
    - Count number of references to definitions.

    ⇒ Size metric: *Number of References*

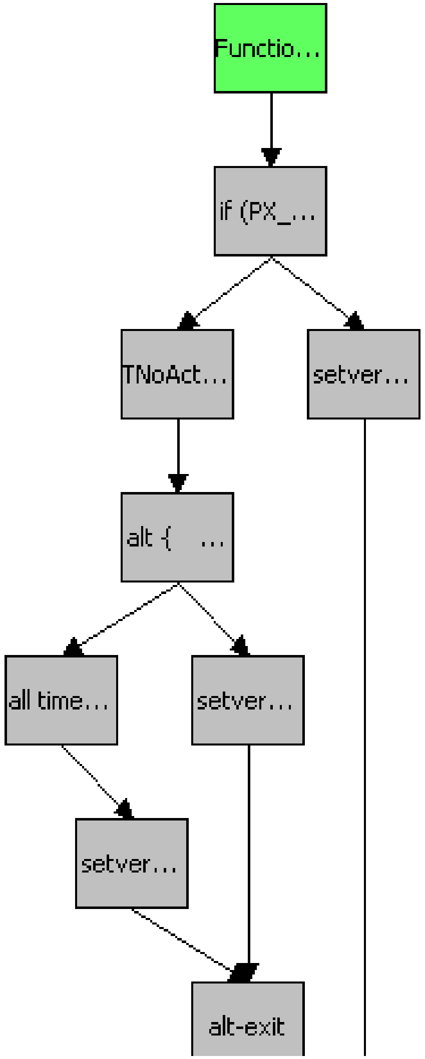
- Goal: Identify error prone TTCN-3 behaviour.
  - Question: "Are there many branches in the behaviour?"
    - Determine number of branches in control flow graph.

    ⇒ Structural complexity metric:

    *Cyclomatic Complexity* v(G):=e-n+2

      - McCabe: *A Complexity Measure*. IEEE Transactions on SE, 1976

e=#edges,
n=#nodes
of control
flow graph G



# Bad Smells in TTCN-3 Test Suites

- Sometimes sophisticated pattern-based approach required, e.g.:
  - Goal: Improve maintainability of TTCN-3 source code.
    - Question:
      "Do local changes require further non-local changes?"
      - Find duplicated code.

⇒ Bad "smells": patterns of inappropriate usage of TTCN-3.

- TTCN-3 smell catalogue:
  - More than 30 TTCN-3 smells:
    - Duplicated code, reference anomalies, violation of coding standards.

# Rule-Based Issue Detection

- Metrics-based:
  - *Number of references to a template* = 0
    ⇒ Remove template.
  - *Cyclomatic complexity* > 10
    ⇒ Extract function.
- Smell-based:
  - Identical actual parameter value
    ⇒ Inline template parameter.
  - Duplicate branches in alt statements
    ⇒ Extract altstep.

# Outline

1. Introduction
2. Metrics / Smells
3. **Refactoring**
4. TRex Tool
5. Related Work
6. 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

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

# Refactoring Description

- Fixed format:
  - Name
  - Summary
  - Motivation
  - Mechanics
  - Example
    - unrefactored
    - refactored

# 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);
  }

}
```

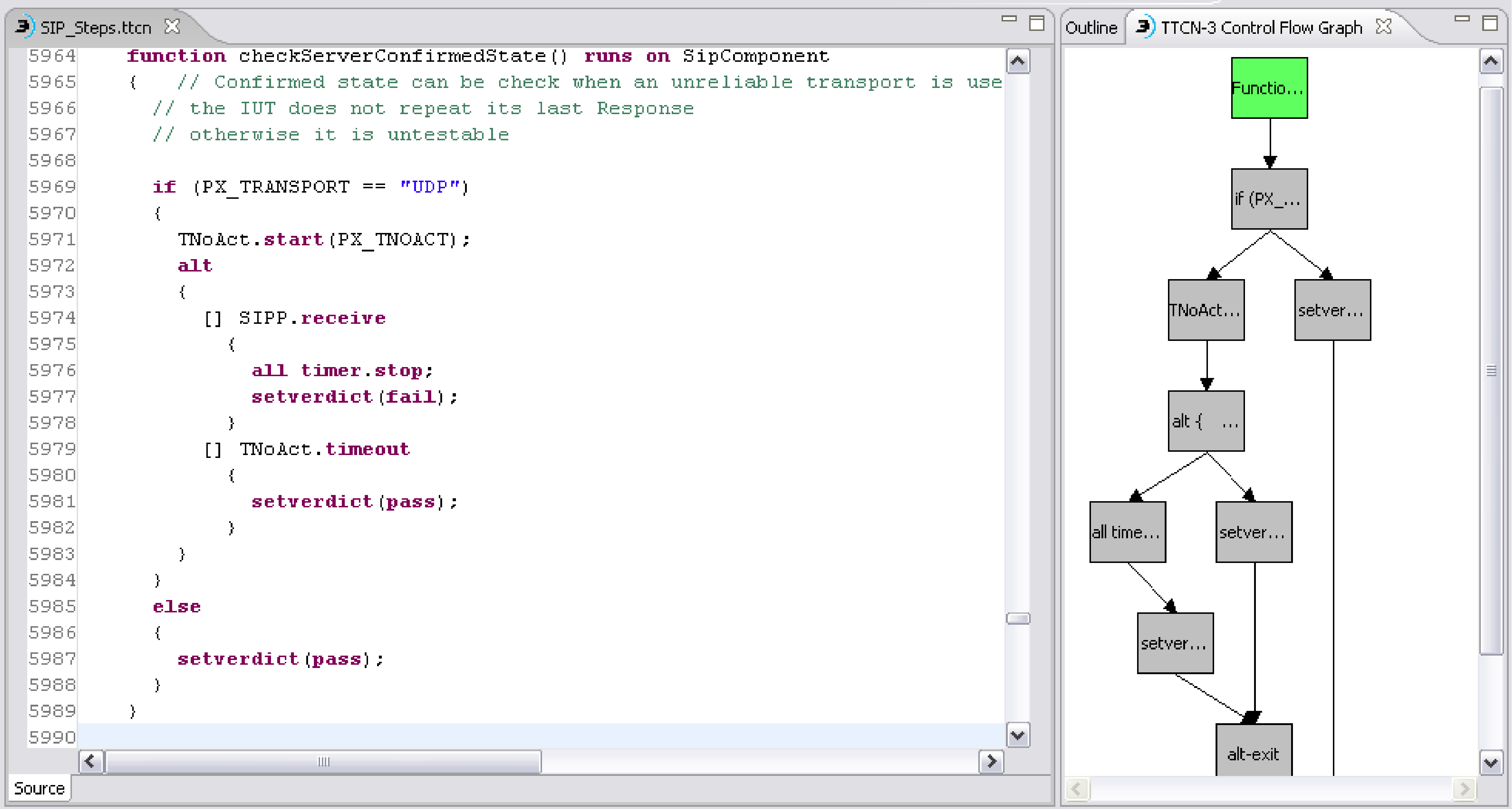
# Outline

1. Introduction
2. Metrics / Smells
3. Refactoring
4. **TRex Tool**
5. Related Work
6. Summary / Outlook

# 4. TRex

- TTCN-3 Refactoring and Metrics tool:
  - Open source plug-in for Eclipse platform,
  - Integrated TTCN-3 development environment,
  - Automated calculation of metrics,
  - Automated detection of smells,
  - Rule-based issue detection,
  - Tool supported refactoring,
  - Visualisation of control flow and call graphs.

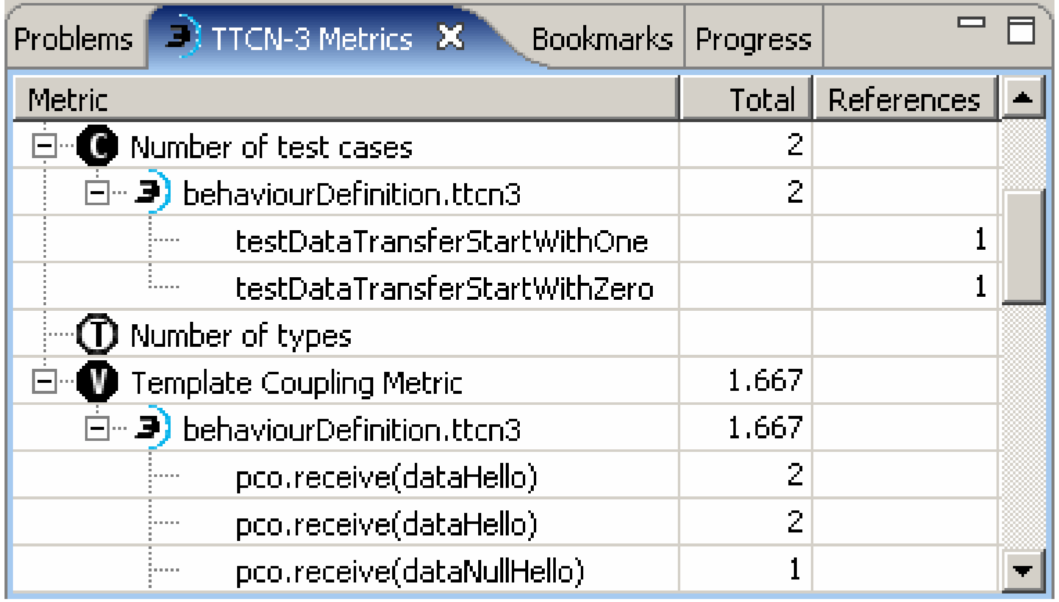
# Visualisation of Control Flow Graph



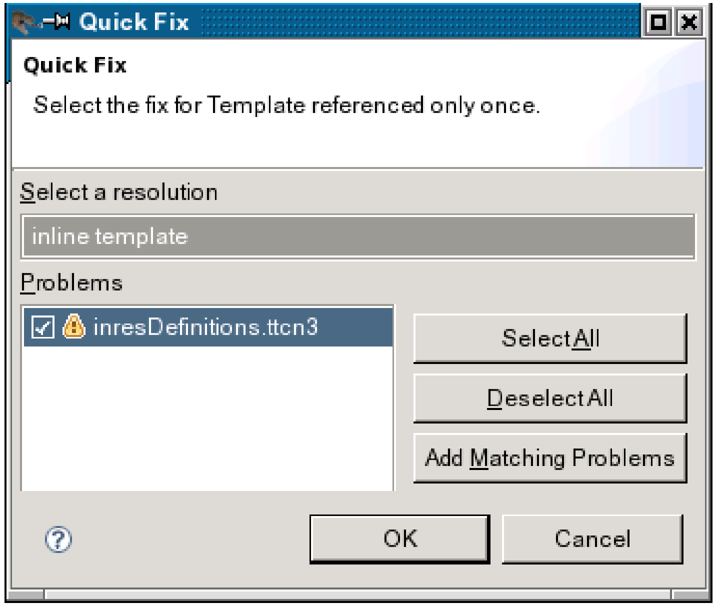
```
function checkServerConfirmedState() runs on SipComponent
{   // Confirmed state can be check when an unreliable transport is use
  // the IUT does not repeat its last Response
  // otherwise it is untestable

  if (PX_TRANSPORT == "UDP")
  {
    TNoAct.start(PX_TNOACT);
    alt
    {
      [] SIPP.receive
        {
          all timer.stop;
          setverdict(fail);
        }
      [] TNoAct.timeout
        {
          setverdict(pass);
        }
    }
  }
  else
  {
    setverdict(pass);
  }
}
```

# Rule-Based Issue Detection



| Metric | Total | References |
|---|---|---|
| Number of test cases | 2 | |
| behaviourDefinition.ttcn3 | 2 | |
| testDataTransferStartWithOne | | 1 |
| testDataTransferStartWithZero | | 1 |
| Number of types | | |
| Template Coupling Metric | 1.667 | |
| behaviourDefinition.ttcn3 | 1.667 | |
| pco.receive(dataHello) | 2 | |
| pco.receive(dataHello) | 2 | |
| pco.receive(dataNullHello) | 1 | |



**Quick Fix**

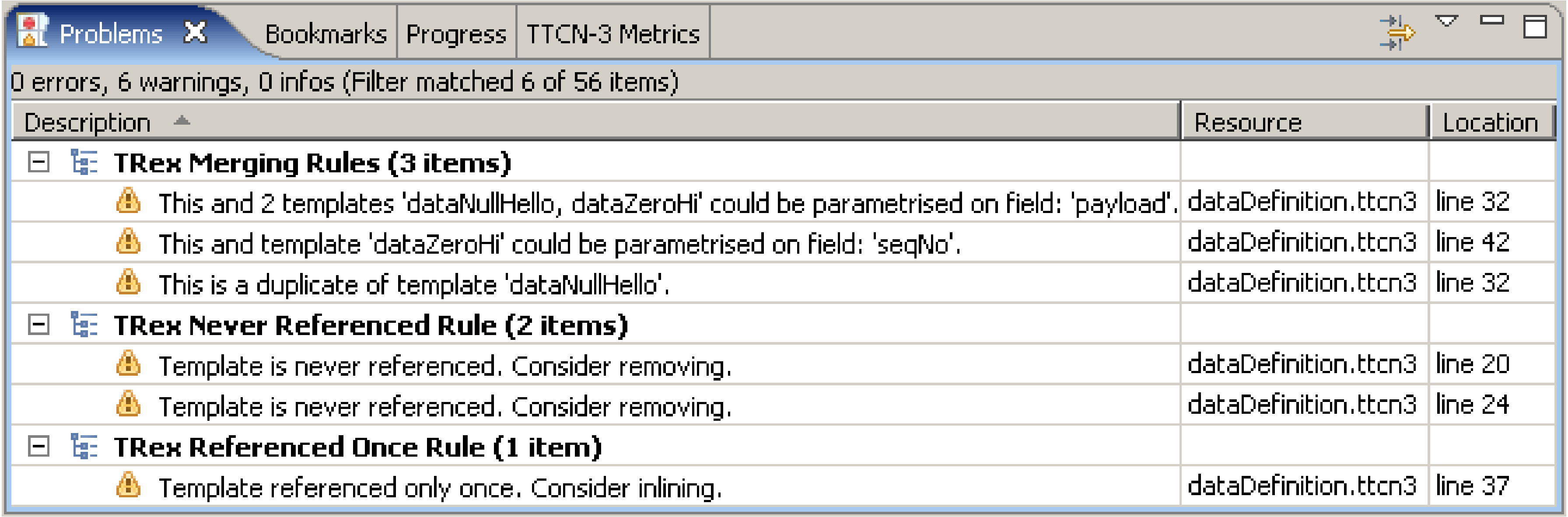
Select the fix for Template referenced only once.

Select a resolution

inline template

Problems

☑ inresDefinitions.ttcn3



0 errors, 6 warnings, 0 infos (Filter matched 6 of 56 items)

| Description | Resource | Location |
|---|---|---|
| **TRex Merging Rules (3 items)** | | |
| This and 2 templates 'dataNullHello, dataZeroHi' could be parametrised on field: 'payload'. | dataDefinition.ttcn3 | line 32 |
| This and template 'dataZeroHi' could be parametrised on field: 'seqNo'. | dataDefinition.ttcn3 | line 42 |
| This is a duplicate of template 'dataNullHello'. | dataDefinition.ttcn3 | line 32 |
| **TRex Never Referenced Rule (2 items)** | | |
| Template is never referenced. Consider removing. | dataDefinition.ttcn3 | line 20 |
| Template is never referenced. Consider removing. | dataDefinition.ttcn3 | line 24 |
| **TRex Referenced Once Rule (1 item)** | | |
| Template referenced only once. Consider inlining. | dataDefinition.ttcn3 | line 37 |

# Application of TRex

- Session Initiation Protocol (SIP) test suite standardised by ETSI:
  - Size:
    - 42397 lines of code (LOC),
    - 528 test cases, 785 functions.
    - 358 templates (5619 LOC).
  - Excerpt of detected issues:
    - 10 unused templates,
    - 22 templates which could be parametrised and merged.
      - Automatic application of refactorings:
        reduction by 393 LOC (7% of template LOC).
    - 119 different duplicate branches in alt statements,
    - 15 behaviours which violate cyclomatic complexity threshold.
      - Related refactorings currently not implemented.

# TTCN-3 Specific Thresholds

- Traditional thresholds for metrics applicable to TTCN-3 as well.
  - E.g.: Cyclomatic complexity v(G)≤10
- Exception: v(control part)
  - Control part used to select test cases to be executed:

```
control {
   if (runRGRT()) {
    if (runRGRTV001()) {
      execute(SIP_RG_RT_V_001());
    };
    if (runRGRTV002()) {
      execute(SIP_RG_RT_V_002());
    };
    …
```

  ⇒ v(control part)=542 for a control part executing 528 test cases.
  - However, linear sequence of guarded executes not very error prone.

  ⇒ Increase v(control part) threshold by number of guarded executes.

# Outline

1. Introduction
2. Metrics / Smells
3. Refactoring
4. TRex Tool
5. **Related Work**
6. Summary / Outlook

# 5. Related Work

- Vega, Schieferdecker:
  *Towards Quality of TTCN-3 Tests.*
  SAM'06: Fifth Workshop on System Analysis and Modelling, 2006
- Schmitt:
  *Automatic Test Generation Based on Formal Specifications – Practical Procedures for Efficient State Space Exploration and Improved Representation of Test Cases.*
  Ph.D. Thesis, University of Göttingen, 2003
- Wu-Hen-Chang, Viet, Batori, Gecse, Csopaki:
  *High-Level Restructuring of TTCN-3 Test Data.*
  Formal Approaches to Software Testing (FATES), 2004
- Deiß:
  *Refactoring and Converting a TTCN-2 Test Suite.*
  Presentation at the TTCN-3 User Conference, 2005

# Outline

1. Introduction
2. Metrics / Smells
3. Refactoring
4. TRex Tool
5. Related Work
6. **Summary / Outlook**

# 6. Summary and Outlook

- Summary:
  - Metrics, smells, rule-based issue detection, refactoring for TTCN-3.
  - TRex tool for automated quality assurance of TTCN-3 test suites.
- Current work:
  - Quality model for test suites.
- Outlook:
  - Implement further refactorings.
  - Simulation to detect non-statically analysable issues.

# Acknowledgements and Further Reading

- Zeiß, Neukirchen, Grabowski, Evans, Baker:
  *Refactoring and Metrics for TTCN-3 Test Suites.*
  Fifth Workshop on System Analysis and Modelling, 2006.
- Zeiß, Neukirchen, Grabowski, Evans, Baker:
  *TRex – An Open-Source Tool for Quality Assurance of TTCN-3 Test Suites.*
  9th International Conference on Quality Engineering in Software Technology, 2006.
- Baker, Evans, Grabowski, Neukirchen, Zeiß:
  *TRex – The Refactoring and Metrics Tool for TTCN-3 Test Specifications.*
  Testing: Academic & Industrial Conference – Practice And Research Techniques, 2006.
- Bisanz: *Pattern-based Smell Detection in TTCN-3 Test Suites.* Master's Thesis, 2006 (to appear).
- Zeiß: *A Refactoring Tool for TTCN-3.* Master's Thesis, 2006.
- Kemnade: *Development of a Semantics-aware Editor for TTCN-3 as an Eclipse Plug-in.* Bachelor's Thesis, 2005.
- Zhao: *Entwicklung eines Parsers für TTCN-3 Version 3 unter Verwendung des Parsergenerators ANTLR.* Bachelor's Thesis, 2005.

- Thank you for your attention!
- Any Questions?

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