

# Qualitätssicherung für Testspezifikationen am Beispiel der standardisierten Testing and Test Control Notation (TTCN-3)

Prof. Dr. Jens Grabowski



Institut für Informatik  
Georg-August-Universität Göttingen

grabowski@cs.uni-goettingen.de



1

## Inhalt

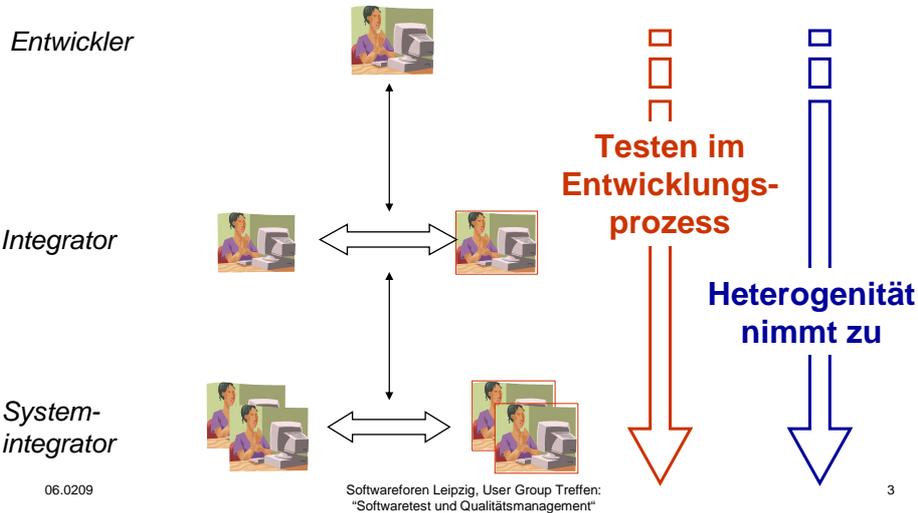
- Einführung
  - Wozu benötigt man eine standardisierte Testsprache?
  - Was ist TTCN-3?
  - Konzepte von TTCN-3
- Qualitätssicherung für TTCN-3-Spezifikationen
  - Vorgehensweise
  - Bewertung von Testreihen
  - Auffinden und Beseitigen von Qualitätsmängeln
  - Implementierung
- Zusammenfassung und Ausblick

06.0209

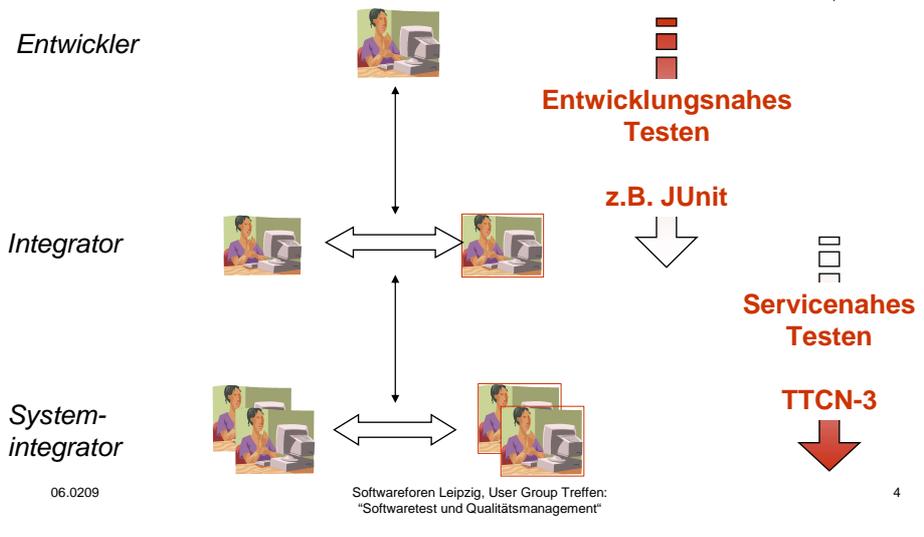
Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

2

# Wozu benötigt man eine standardisierte Testsprache?



# Wozu benötigt man eine standardisierte Testsprache?



## Wozu benötigt man eine standardisierte Testsprache?



- Eine standardisierte Testsprache
  - verbessert die Kommunikation
    - zwischen Entwicklern und Testern
    - mit dem Kunden
  - verbessert die Transparenz des Testprozesses:
    - eine Testsprache für alle Abteilungen
    - vermeidet proprietäre Testsprachen
  - verringert die Testkosten:
    - Schulungskosten
    - Verwendung von kommerziellen Testlösungen (mehrerer Anbieter)

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

5

## Was ist TTCN-3?



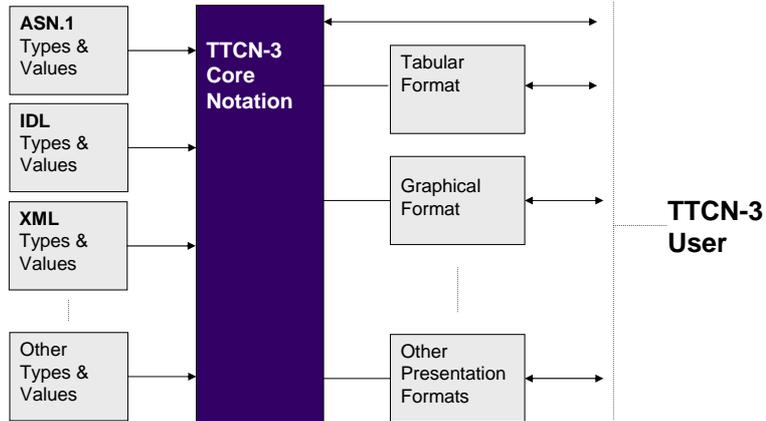
- **Die** standardisierte (Black-Box) Testspezifikations- und Testimplementierungssprache.
- TTCN-3 (= Testing und Test Control Notation version 3)
  - wurde beim European Telecommunications Standards Institute (ETSI) von 1999 – 2001 entwickelt.
  - wird seit 2001 beim ETSI kontinuierlich gepflegt und weiterentwickelt.
  - basiert auf Erfahrungen mit früheren TTCN Versionen.
- Benutzbar für alle Arten des Black-Box Testens von reaktiven und verteilten Systemen
  - Mobile (Telecom) Systeme: ISDN, GSM, UMTS, WiMAX
  - Internet: IPv6
  - Automotive: AUTOSAR

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

6

# Was ist TTCN-3?

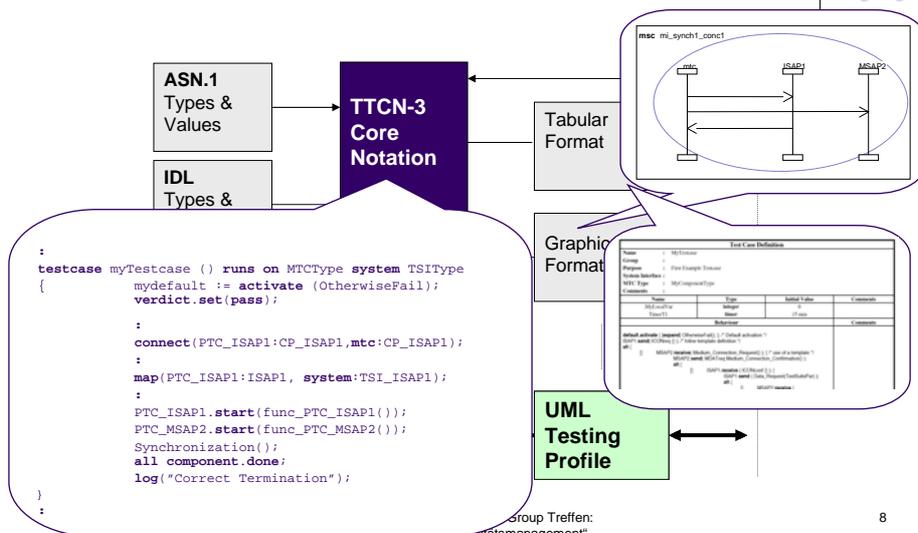


06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

7

# Was ist TTCN-3?



Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

8

## Was ist TTCN-3?



- Europäischer Standard (ES) in 10 Teilen
  - ES 201 873-1: TTCN-3 Core Language
  - ES 201 873-2: TTCN-3 Tabular Presentation Format (TFT)
  - ES 201 873-3: TTCN-3 Graphical Presentation Format (GFT)
  - ES 201 873-4: TTCN-3 Operational Semantics
  - ES 201 873-5: TTCN-3 Runtime Interface (TRI)
  - ES 201 873-6: TTCN-3 Control Interface (TCI)
  - ES 201 873-7: Using ASN.1 with TTCN-3
  - ES 201 873-8: Using IDL with TTCN-3
  - ES 201 873-9: Using XML with TTCN-3
  - ES 201 873-10: Documentation Comment Specification

## Konzepte von TTCN-3



- Black-box Testen mit TTCN-3
- Verteilte TTCN-3 Testkonfigurationen
- TTCN-3 Implementierung
- Weitere Konzepte

# Black-box Testen mit TTCN-3

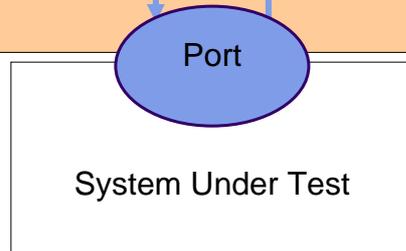


## TTCN-3 Test Case

Port.send(Stimulus)

Port.receive(Response)

- Assignment of a Test Verdict

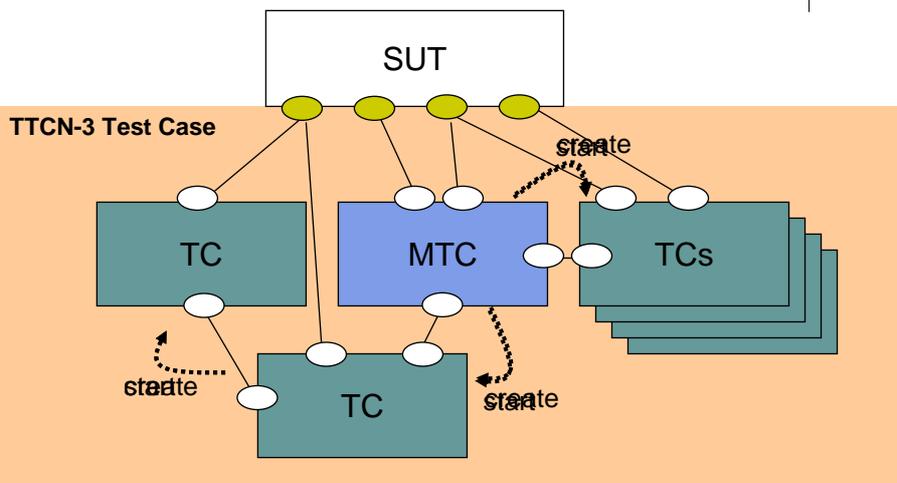


06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

11

# Verteilte TTCN-3 Testkonfigurationen

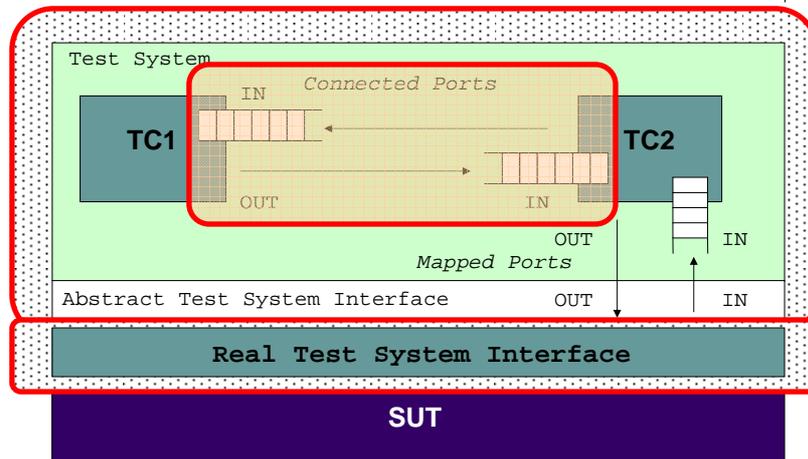


06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

12

# TTCN-3 Implementierung



06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

13

# Konzepte von TTCN-3



- Black-box Testen mit TTCN-3
- Verteilte TTCN-3 Testkonfigurationen
- TTCN-3 Implementierung
- Weitere Konzepte
  - Umfangreiches Datentypsysteem
  - Ausgefeiltes System zur Beschreibung von Testdaten (Templates mit Matchingmechanismen)
  - Default-Verhalten
  - Unterstützt verschiedene Kommunikationsmechanismen

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

14

# TTCN-3 Kernsprache



- Beispiel:

```
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“ einer typischen Programmiersprache
- **Qualitätsprobleme wie anderer Quellcode!**

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

15

# Umfang von TTCN-3 Testreihen



- Motorola (interne) riesige „legacy“ Testreihen
  - Migration zu TTCN-3
  - Automatische Konvertierung einer UMTS Testreihe
    - **60.000 Lines of Code (LOC)**
    - Schwer zu lesen, zu verstehen und zu pflegen
- Standardisierte Testreihen (ETSI)
  - SIP (ETSI TS 102 027-3 - v4.2.5): **61.990 LOC**
  - IPv6 Core Protocol (ETSI TS 102.516 - v3.1.1): **67.580 LOC**
  - 3GPP Benchmark: **73.130 LOC**

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

16

# Inhalt



- Einführung
  - Wozu benötigt man eine standardisierte Testsprache?
  - Was ist TTCN-3?
  - Konzepte von TTCN-3
- Qualitätssicherung für TTCN-3-Spezifikationen
  - Vorgehensweise
  - Bewertung von Testreihen
  - Auffinden und Beseitigen von Qualitätsmängeln
  - Implementierung
- Zusammenfassung und Ausblick

# Vorgehensweise: Qualitätssicherung für TTCN-3 Spezifikationen



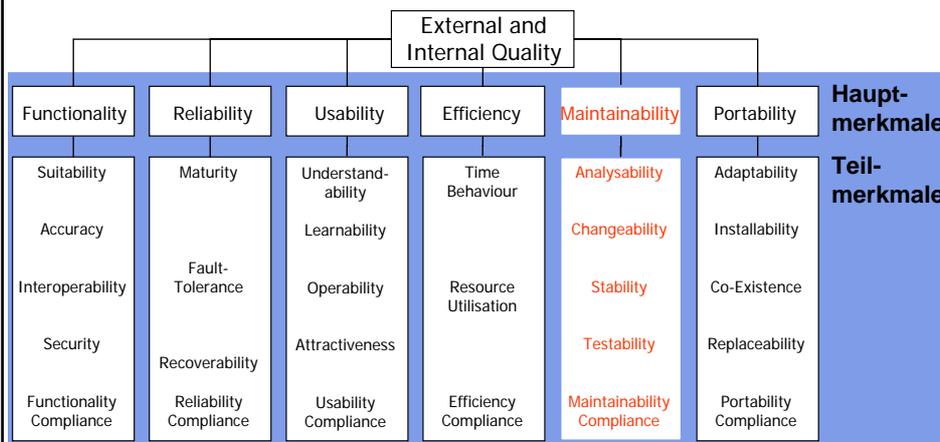
- Bewertung von Testreihen → *Qualitätsmodell*
- Auffinden von Qualitätsmängeln → *Metrik- und Code Smell-basiert*
- Beseitigen von Qualitätsmängeln → *Refactoring*

# Bewertung von Testreihen - Qualitätsmodelle

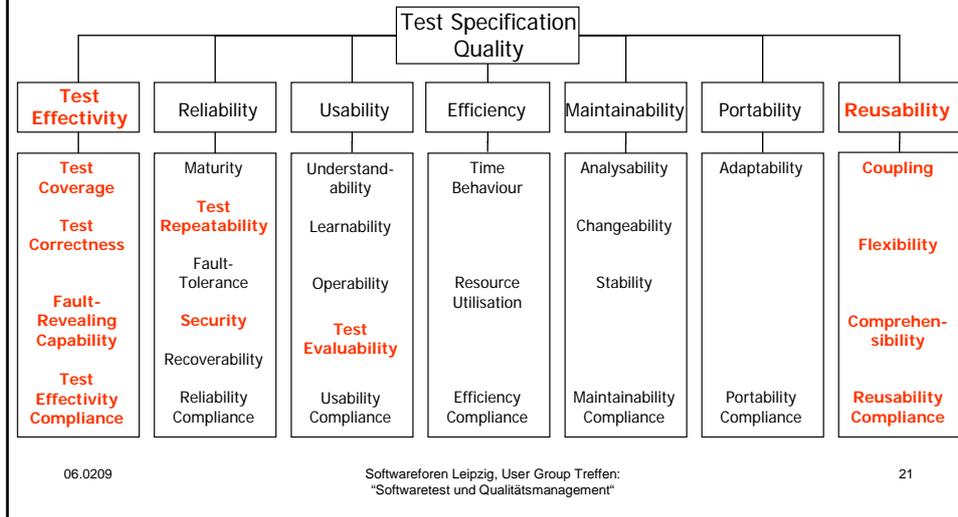


- Objektive Beurteilung von Software-Qualität,
  - Objektive Zielvorgaben für Software-Qualität.
  - ISO 9126-1:  
Software Engineering – Product Quality – Quality Model
    - Qualitätsmodelle für
      - Interne Qualität,
      - Externe Qualität,
      - Quality in Use.
- Qualität setzt sich aus einzelnen Merkmalen sowie ggf. weiteren Teilmerkmalen zusammen.

# Das ISO 9126 Modell für interne und externe Qualität



# Ein Qualitätsmodell für Testspezifikationen



# Instantiierung von Qualitätsmodellen



- Qualitätsmodell abstrahiert von
  - Testspezifikationsprache,
  - projekt-spezifischen Anforderungen.
 ⇒ **Instantiierung nötig!**
  
- ISO 14598: Software Engineering – Product Evaluation
  1. Qualitätsmodell erstellen,
  2. Metriken für Qualitätsmerkmale festlegen,
  3. Grenzwerte für Metriken festlegen,
  4. Gewichtung der Qualitätsmerkmale.

# Beispiel: TTCN-3 Metriken für Qualitätsmerkmal Maintainability



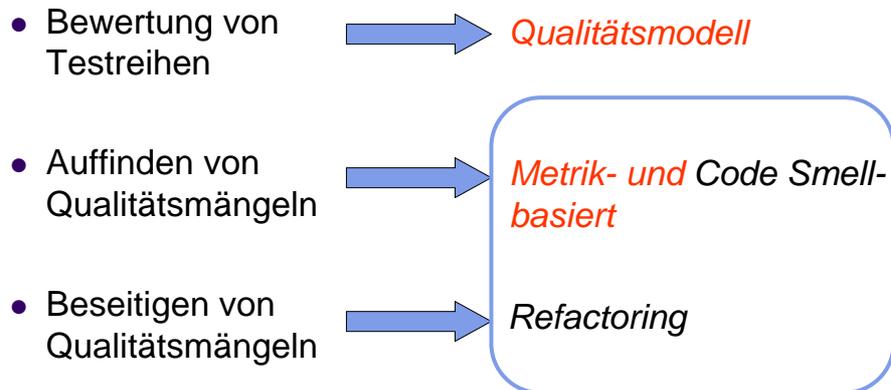
- Maintainability:
  - Analysability:
    - *complexity violation*  $:= 1 - \frac{\sum \text{Komplexe Verhaltenseinheiten}}{\sum \text{Verhaltenseinheiten}}$
  - Changeability:
    - *code duplication*  $:= 1 - \frac{\sum \text{Duplizierte Quelltexteinheiten}}{\sum \text{Quelltexteinheiten}}$
  - Stability:
    - *parameter reassignment*  $:= 1 - \frac{\sum \text{out und inout Formalparameter}}{\sum \text{Formalparameter}}$
- Metrikintervalle:  
0,0 (= schlechteste Qualität) bis 1,0 (= beste Qualität).

# Anwendung des Qualitätsmodells



Metrik	SIP v2.20	SIP v2.24	SIP v3.01	SIP v3.06
Testfälle	1068	1068	1412	1412
Verhaltenseinheiten	1961	1971	2360	2369
Verhaltenseinheiten mit zyklomatischer Komplexität > 10	27	30	51	51
Zweige in alt-Anweisungen	1900	1958	2482	2534
Duplizierte Zweige in alt-Anweisungen	1435	1471	1849	1879
Formalparameter	3175	3224	5062	5084
out und inout Formalparameter	1237	1244	1617	1628
Analysability: <i>complexity violation</i> (zyklomatische Komplexität >10)	0.99	0.98	0.98	0.98
Changeability: <i>code duplication</i> (bzgl. Zweigen in alt-Anweisungen)	0.25	0.25	0.26	0.26
Stability: <i>parameter reassignment</i>	0.61	0.61	0.68	0.68

## Vorgehensweise: Qualitätssicherung für TTCN-3 Spezifikationen



06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

25

## Definition: Code Smell & Refactoring



- **Code smell:**  
"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*
- **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*

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

26

# TTCN-3 Code Smells



- TTCN-3 Code Smells:
  - *Muster für die unsachgemäße Benutzung von TTCN-3.*
  - Code Smells lassen sich durch Refactoring verbessern.
    - Zeiss, Neukirchen, Grabowski, Evans, Baker: *Refactoring and Metrics for TTCN-3 Test Suites*. SAM-Workshop, 2006.
  - Per Definition keine Code Smells:
    - Syntax Fehler,
    - Verstöße gegen die statische Semantik,
    - Fehler in der Testfall Logik.
  - TTCN-3 Code Smells geben nur Hinweise auf Qualitätsprobleme
    - Was als TTCN-3 Code Smell angesehen werden soll, ist projektabhängig.

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

27

# TTCN-3 Code Smell Katalog



- Für TTCN-3 Code Smells wurde ein strukturierter Katalog angelegt.
- Bisher wurden 38 TTCN-3 Code Smells identifiziert, die folgende Aspekte abdecken:
  - **Duplicated Code**, z.B. *Duplicate Alt Branches*
  - **References**, z.B. *Singular Component Variable/Const./Timer*
  - **Parameters**, z.B. *Constant Actual Parameter Value*
  - **Complexity**, z.B. *Complex Conditional*
  - **Default Anomalies**, z.B. *Activation Asymmetry*
  - **Test Behaviour**, z.B. *Missing Verdict*
  - **Test Configuration**, z.B. *Idle Parallel Test Component*
  - **Coding Standards**, z.B. *Magic Values*
  - **Data Flow Anomalies**, z.B. *Unused Variable Definition*
  - **Miscellaneous**, z.B. *Over-specific Runs On*

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

28

## Definition: Code Smell & Refactoring



- Code smell:  
“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

- 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.”

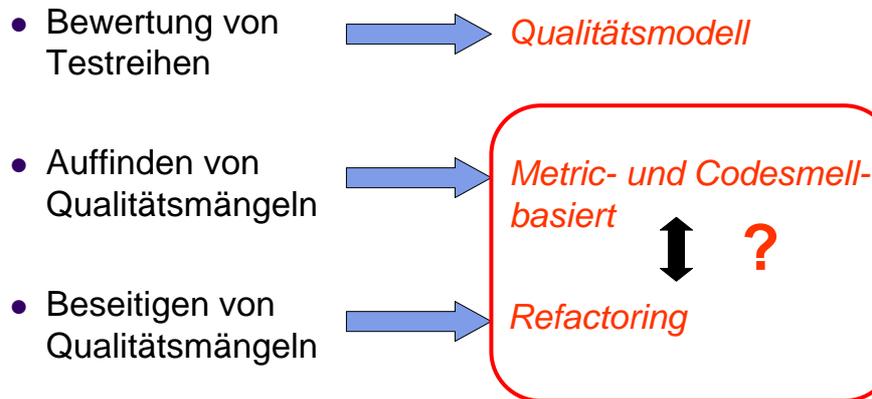
Fowler: *Refactoring – Improving the Design of Existing Code.*  
Addison-Wesley, 1999

## TTCN-3 Refactoring Katalog



- 28 Refactorings von Fowler, die auf TTCN-3 anwendbar sind.
- 20 TTCN-2 spezifische Refactorings.
- Katalog Struktur: Refactorings zur Verbesserung von
  - Testverhalten (20 Refactorings):
    - *Extract Altstep*,
    - ...
  - der allg. Struktur der Testreihe (22 Refactorings):
    - *Extract Module*,
    - ...
  - Datenbeschreibungen (6 Refactorings):
    - *Inline Template Parameter*,
    - ...

## Vorgehensweise: Qualitätssicherung für TTCN-3 Spezifikationen



06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

31

## Auffinden und Beseitigen von Qualitätsmängeln



- **Metrik-basiert:**
  - Anzahl der Referenzen auf ein Template = 0  
⇒ Lösche Template
  - Anzahl der Referenzen auf ein Template = 1  
⇒ Definition des Templates bei dessen Benutzung
- **Code Smell-basiert:**
  - Verwendung von Templates mit identischen Parameterwerten  
⇒ Parameter in das Template integrieren
  - Mehrere Templates unterscheiden sich nur in einem Wert  
⇒ Parameterisiere Template

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

32



TTCN-3

## Implementierung: TRex



- TTCN-3 Refactoring and Metrics Tool (TRex):
  - Open Source Plug-In für die Eclipse-Plattform
  - Integrierte TTCN-3 Entwicklungsumgebung
  - Automatische Berechnung von Metriken
  - Automatische Detektion von TTCN-3 Code Smells
  - Regel- und Metrik-basierte Erkennen von Qualitätsproblemen
  - Werkzeug-basiertes Refactoring
  - Visualisierung von Kontrollfluss- und (Funktions-)Aufrufgraphen.

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

33



TTCN-3

## Visualisierung von Kontrollflussgraphen



The screenshot displays the TRex tool interface. On the left, a code editor shows the source code for a function named `checkServerConfirmedState()`. The code includes comments and logic for handling transport types (UDP) and timeouts. On the right, a control flow graph (CFG) visualizes the execution flow of the code. The graph starts with a function node, followed by a conditional node `if (PX_...)`. This leads to two parallel paths: one for `TNoAct...` and another for `setver...`. The `TNoAct...` path includes an `alt` block with two branches: `SIPP.receive` (which contains `all timer.stop;` and `setverdict(fail);`) and `TNoAct.timeout` (which contains `setverdict(pass);`). Both paths eventually lead to an `alt-exit` node.

```
5964 function checkServerConfirmedState() runs on SipComponent
5965 { // Confirmed state can be check when an unreliable transport is use
5966   // the IUT does not repeat its last Response
5967   // otherwise it is untestable
5968
5969   if (PX_TRANSPORT == "UDP")
5970   {
5971     TNoAct.start(PX_TNO&ACT);
5972     alt
5973     {
5974       [] SIPP.receive
5975       {
5976         all timer.stop;
5977         setverdict(fail);
5978       }
5979       [] TNoAct.timeout
5980       {
5981         setverdict(pass);
5982       }
5983     }
5984   }
5985   else
5986   {
5987     setverdict(pass);
5988   }
5989 }
5990
```



## Regel- und Metrik-basierte Erkennung von Qualitätsproblemen



The screenshot shows the TTTech Metrics tool interface. The main window displays a table of metrics:

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	

A 'Quick Fix' dialog box is open, showing a list of problems with a 'Select All' button. Below the dialog, a 'Problems' window shows a list of detected issues:

Description	Resource	Location
<b>TRex Merging Rules (3 items)</b>		
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
<b>TRex Never Referenced Rule (2 items)</b>		
Template is never referenced. Consider removing.	dataDefinition.ttcn3	line 20
Template is never referenced. Consider removing.	dataDefinition.ttcn3	line 24
<b>TRex Referenced Once Rule (1 item)</b>		
Template referenced only once. Consider inlining.	dataDefinition.ttcn3	line 37

35



## Anwendung von TRex



- Session Initiation Protocol (SIP) Testreihe (standardisiert von ETSI):
  - Größe:
    - 42397 lines of code (LOC),
    - 528 Testfälle, 785 Funktionen,
    - 358 Templates (5619 LOC).
  - Auszug von gefundenen Qualitätsproblemen:
    - 10 unbenutzte Templates,
    - 22 Templates, die man parameterisieren und zusammengeführten könnte.
      - Automatische Anwendung der zugehörigen Refactorings führten zu einer Reduktion der Testreihe um 393 LOC (7% der Template LOC).
    - 119 verschiedene mehrfach duplizierte alt-Verzweigungen.
    - 15 Verhalten, die gegen die McCabe-Komplexität verstoßen.
      - Refactorings zur Beseitigung dieser Qualitätsprobleme sind noch nicht implementiert.

06.0209

Softwareforen Leipzig, User Group Treffen:  
"Softwaretest und Qualitätsmanagement"

36

# Inhalt



- Einführung
  - Wozu benötigt man eine standardisierte Testsprache?
  - Was ist TTCN-3?
  - Konzepte von TTCN-3
- Qualitätssicherung für TTCN-3-Spezifikationen
  - Vorgehensweise
  - Bewertung von Testreihen
  - Auffinden und Beseitigen von Qualitätsmängeln
  - Implementierung
- Zusammenfassung und Ausblick

# Zusammenfassung



- Übersicht über TTCN-3
- Methodik zur Bewertung und Verbesserung der Qualität von TTCN-3. Die Methodik basiert auf:
  - einem *Qualitätsmodell* zur Bewertung von Testreihen,
  - *Metriken* und *TTCN-3 Code Smells* zur Erkennung von Qualitätsproblemen und
  - *Refactoring* zur Qualitätsverbesserung.
- Methodik wurde im TRex-Werkzeug implementiert.
- An Beispielen wurde die Anwendbarkeit dieser Methodik gezeigt.

## Ausblick



- Detektion und Verbesserung von Qualitätsproblemen, die sich nicht statisch entdecken lassen.
- Qualität von Testreihen, die nicht-funktionale Anforderungen (u.a. Realzeitaspekte) messen (TEMEA-Projekt).
- Enge Zusammenarbeit mit ETSI im Bereich der Qualitätssicherung für standardisierte TTCN-3-Testreihen.

- Vielen Dank für Ihre Aufmerksamkeit!
- Haben Sie Fragen?



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