Quality Engineering for Test Artifacts

Jens Grabowski, Benjamin Zeiss
University of Göttingen, Germany

Motivation

- Test specification complexity is growing
- Current TTCN-3 test suites:
  - SIP (ETSI TS 102 027-3 - v4.3.5): 61,990 LOC
  - IPv6 Core Protocol (ETSI TS 102.516 - v3.1.1): 67,580 LOC
  - 3GPP / LTE (STF 160): 71,687 LOC
- Quality Assurance (QA) of test specifications is necessary!
- Quality assurance classification:
  - Organizational Quality Assurance (infrastructure and management)
  - Constructive Product Quality Assurance (avoiding issues, preventative)
  - Analytical Product Quality Assurance (eliminating issues, reactive)
- How can we apply QA to test specification development?

TAROT 09

- Testing and Test Control Notation version 3
- Language for specifying and implementing distributed tests.
- Standardized by:
  - European Telecommunications Standards Institute (ETSI),
  - International Telecommunication Union (ITU).
- History:
  - Standardization 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,
  - Uses standardized test suites against their implementation.
- Today:
  - TTCN-3 not only used in telecommunication domain, but for Internet, Service-Oriented Architectures, Automotive, ...

TTCN-3 (2/2)

- 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

Agenda

- Organizational Quality Assurance
- Constructive Product Quality Assurance
- Analytical Product Quality Assurance
- Tools
- Summary / Conclusion
Organizational Quality Assurance

**Assurance of infrastructure and management quality**

**Infrastructure quality:**
- Configuration management
- Defect management
- ...

**Management quality:**
- Test process model
- Process improvement models

---

**Organizational Quality Assurance: Test process model (1/3)**

- Describes activities that need to happen in development
- On what subject something needs to happen
- When something happens
- Why it needs to happen

**Example: ISQTB fundamental test process:**

![Test process model](image)

---

**Organizational Quality Assurance: Test process model (2/3)**

- Example: ISQTB fundamental test process (continued)

  - Test planning and test control
    - Testing approach, resource planning, test strategy, schedule, metrics definition for monitoring and control
    - Test plan document: IEEE 829 Test Plan Specification
    - Test analysis and test design
    - Decision what to test, how much effort to spend, what types of testing, tools, risk analysis
    - Test specification: IEEE 829 Test Design Specification
    - Document reviews
  - Test implementation and test execution
    - Definition of high-level and concrete test cases
    - TTCN-3

---

**Organizational Quality Assurance: Test process model (3/3)**

- Example: ISQTB fundamental test process (continued)

  - Evaluation of exit criteria and reporting
    - Test summary report (test progress / test conclusion)
    - Exit criteria from the planning phase satisfied?
  - Closure
    - Wrap-up reports
    - Archiving of test documents and test data

---

**Organizational Quality Assurance: Process improvement models**

**Generic process improvement models**
- Capability Maturity Model integrated (CMMI)
- SPICE (ISO 15504)

**Test process improvement models**
- Test Maturity Model integrated (TMMi)
- Test Process Improvement (TPI)
- Critical Testing Processes (CTP)
- Systematic Test and Evaluation Process (STEP)
Organizational Quality Assurance:
Process improvement models: TMMi (1/3)

- TMMi = Test Maturity Model integrated
- Originally developed by Ilene Burnstein at Illinois Institute of Technology since 1996
- Complementary to CMMI
- TMMi reference model describes level 1-3 in detail
- Can be mapped to other models like TPI

Organizational Quality Assurance:
Process improvement models: TMMi (2/3)

- Staged model
  - Level 1 (Initial):
    - Chaos, ad-hoc debugging
  - Level 2 (Managed):
    - Test policy and strategy, test planning, test monitoring and control, test design and execution, test environment
    - Main objective: does the product satisfy the specified requirements? (Functional Testing)
  - Level 3 (Defined):
    - Test organization, test training program, test life-cycle and integration, non-functional testing, peer reviews
    - Testing at an early project stage
    - More rigorous process descriptions

Organizational Quality Assurance:
Process improvement models: TMMi (3/3)

- Staged model (continued)
  - Level 4 (Management and Measurement):
    - Test measurement, software quality evaluation, advanced peer reviews
    - Measurements to support decision making
  - Level 5 (Optimization):
    - Defect prevention, test process optimization, quality control
    - Continuous test process improvement based on quantitative understanding

Agenda

- Organizational Quality Assurance
- Constructive Product Quality Assurance
- Analytical Product Quality Assurance
- Tools
- Summary / Conclusion

Constructive Product Quality Assurance

- Training and technology certifications
- High-level testing languages
- Best practices:
  - Test development guidelines
  - Test patterns
  - Tooling

Constructive Product Quality Assurance

- Training and technology certifications
- High-level testing languages
- Best practices:
  - Test development guidelines
  - Test patterns
  - Tooling
Constructive Product Quality Assurance:
Training and technology certifications (1/2)

- Quality Assurance Institute (QAI)
  - http://www.qaiworldwide.org
  - Certified Software Test Engineer (CSTE)
  - Certified Software Quality Analyst (CSQA)

- American Society for Quality (ASQ)
  - http://www.asq.org
  - Software Quality Engineer (CSQE)
  - Quality Inspector (CQI)

- International Institute for Software Testing (IIST)
  - http://www.iist.org
  - Certified Software Test Professional (CSTP)
  - Certified Test Manager (CTM)

Constructive Product Quality Assurance:
Training and technology certifications (2/2)

- International Software Qualifications Testing Board (ISQTB)
  - http://www.isqtb.org
  - Certified Tester Foundation Level (CTFL)
  - Certified Tester Advanced Level (CTAL)
  - Certified Tester Expert Level (in preparation)

- German Testing Board (GTB)
  - http://www.german-testing-board.de
  - TTCN-3 Certificate
    - http://german-testing-board.de/en/ttcn3_certificates.htm

Constructive Product Quality Assurance

- Training and technology certifications

  High-level testing languages

  Best practices:

    - Test development guidelines
    - Test patterns
    - Tooling

Constructive Product Quality Assurance

- Training and technology certifications

  High-level testing languages

  Best practices:

    - Test development guidelines
    - Test patterns
    - Tooling

Constructive Product Quality Assurance

- Training and technology certifications

  High-level testing languages

  Best practices: test development guidelines

    - Naming conventions
      - Example: module names start with an upper-case letter
    - Code formatting guidelines
      - Example: spaces instead of tabs
    - Code documentation guidelines
      - Example: all test cases must have a @desc and @param description
    - Code style guidelines
      - Example: no labels or goto statements
    - Usage of test patterns
Constructive Product Quality Assurance

- Training and technology certifications
- High-level testing languages
- Best practices:
  - Test development guidelines
  - Test patterns
  - Tooling

Constructive Product Quality Assurance: Best practices: test patterns

- Identify and document general and proven solutions for commonly occurring problems
- Provide a common vocabulary for these solutions
- Catalog with a fixed notation
- Exemplified for TTCN-3: classification proposal for TTCN-3 specification

Best practices: test patterns

- Test development guidelines
- Test patterns
- Tooling

Constructive Product Quality Assurance: Best practices: test patterns – example (1/3)

<table>
<thead>
<tr>
<th>Name</th>
<th>Timer on transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Behavior</td>
</tr>
<tr>
<td>Testing phases</td>
<td>Specification</td>
</tr>
<tr>
<td>Testing goals</td>
<td>All</td>
</tr>
<tr>
<td>Application domain</td>
<td>Any</td>
</tr>
<tr>
<td>Intend</td>
<td>Avoid deadlock situation when transmitting data from test component</td>
</tr>
<tr>
<td>Context</td>
<td>For the testing of reactive systems tested typically via interfaces</td>
</tr>
<tr>
<td>Parameter</td>
<td>Timer duration</td>
</tr>
</tbody>
</table>

Constructive Product Quality Assurance: Best practices: test patterns – example (2/3)

<table>
<thead>
<tr>
<th>Roles</th>
<th>test component, source port, destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detailed description</td>
<td>After calling a method from a test component or sending a message on a given port, a timer should be started to avoid deadlock in case the SUT does not reply to the function call or to the transmitted message.</td>
</tr>
</tbody>
</table>

Example

```plaintext
function timedSend (template <outMessageType> <outMsg>, <OutPortType> <outPort>, timer <t>, template <inMessageType> <inMsg>, <InPortType> <inPort>) returns verdicttype {
    <outPort>.send(<outMsg>);
    <t>.start;
    alt {
        <inPort>.receive(<inMsg>) {
            <t>.stop;
            return pass;
        }
        <t>.timeout {
            return fail;
        }
    }
}
```

Constructive Product Quality Assurance: Best practices: test patterns – example (3/3)

<table>
<thead>
<tr>
<th>Consequences</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related patterns</td>
<td>Default pattern</td>
</tr>
<tr>
<td>Known uses</td>
<td>Protocol testing</td>
</tr>
</tbody>
</table>

Constructive Product Quality Assurance

- Training and technology certifications
- High-level testing languages
- Best practices:
  - Test development guidelines
  - Test patterns
  - Tooling
**Constructive Product Quality Assurance:**

Best practices: tooling (1/2)

- Are the tools standards conformant?
- Can I switch tool vendors easily?
- Do the tools interoperate with other tools?
- Tool efficiency?
- Integrated Development Environment?
- Documentation generator?

Best practices: tooling (2/2)

- Reporting?
- Analytical support?
- Support for quality assurance?
- Version control?
- Defect management system?
- Build automation / continuous integration?

**Agenda**

- Organizational Quality Assurance
- Constructive Product Quality Assurance
- Analytical Product Quality Assurance
- Tools
- Summary / Conclusion

**Analytical Product Quality Assurance**

- Code Review
- Guideline Checking
- Test Metrics
- Smell Detection
- Simulation
- Model-Based Analysis
- Guideline Checking
- Test Metrics
- Small Detection
- Code Review

Improvement: test refactoring (and debugging/correction)

**Analytical Test Quality Assurance**

- Static Testing
  - With Computer
  - Without Computer
    - Guideline Checking
    - Test Metrics
    - Small Detection
    - Code Review

Improve testing: refactoring
Analytical Product Quality Assurance:
Static testing: code reviews (2/2)

- Review types:
  - Informal review
    - Reviewers are asked to deliver remarks.
    - No meeting, no formal documentation.
  - Walkthrough
    - Author presents artifact to the reviewers in a meeting. Remarks based on the presentation.
  - Technical review
    - Does the test conform to its specification?
    - Fitness for its purpose, Standards compliance.
    - Meeting, unanimous review result.
  - Inspection
    - Formal with focus on possible defect, document quality, product and process quality.
    - Strict Agenda.

Analytical Product Quality Assurance:
Static testing: guideline checking

- Semantic analysis / type checking
  - Float assignment to an integer variable,
  - Reading of undefined variable,
  - Call to undefined function,
  - Number of arguments mismatch,
  - Duplicate functions/test case names, etc.
  - Should be done by the compiler

- Naming conventions / style guidelines
  - Test case should start with prefix "TC_".
  - Every test case must have a T3Doc comment.
  - Log messages must have a certain format.
  - Types should be defined in alphabetic order.
  - Variables should be declared first in a test case, etc.

Analytical Product Quality Assurance:
Static testing: test metrics (1/11)

"When you measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science."  
Lord Kelvin, 1891

"You cannot control what you cannot measure."  
Tom DeMarco, 1982

Analytical Product Quality Assurance:
Static testing: test metrics (2/11)

- Metric: measurement scale + measurement
  - Mapping of a software property into a numerical value or symbol of the measurement scale

- Vast amount of metrics have been suggested:
  - Counting (Number of XXX, LOC)
  - Cyclomatic complexity
  - Coverage
  - Coupling
  - …
Analytical Product Quality Assurance: Static testing: test metrics (3/11)

Metrics classification (Fenton, Pfleeger):

- Software metrics
  - Processes
  - Products
  - Resources
  - External Attributes
  - Internal Attributes
  - Size Metrics
  - Structural Metrics

Analytical Product Quality Assurance: Static testing: test metrics (4/11)

The inconvenient truth:
- Metrics are not universal!
- Metric thresholds are not universal!

Analytical Product Quality Assurance: Static testing: test metrics (5/11)

You are often interested in exactly the opposite of what you measure:
- 90% tests with verdict “pass”
  - Why are 10% of the tests failing or inconclusive? Do the 10% reflect the severity of the negative test results?
- 90% specification coverage
  - Why is 10% of the specification untested? Hard to test?
- Number of test cases
  - How can I select test cases (maximize coverage, minimize test executions)?

Metrics tell you what happened, but not why!

Analytical Product Quality Assurance: Static testing: test metrics (6/11)

What should be measured then???
- A quality model is required!

A quality model for a product provides:
- The main quality characteristics of a product.
- How these quality characteristics are subdivided.

ISO 9126-1:
- Software engineering – Product quality – Quality Model
  - Quality models
    - Internal quality
    - External quality
    - Quality in use.

Quality is composed of discrete characteristics, which may be structured into further sub-characteristics.

Analytical Product Quality Assurance: Static testing: test metrics (7/11)

Reoccurring problems with metric definitions:
- The metric is unbelievable. People don’t understand it.
- The metric is not defined clear or it is ambiguous or non-objective.
- The metric is not repeatable.
- The metric is not comparable to other measurements.
- People can fool the metric to make it look good.
- It is unclear what good and bad values for the metric are (thresholds).
- The metric is “just” interesting and not measured for a reason.

Analytical Product Quality Assurance: Static testing: test metrics (8/11)

<table>
<thead>
<tr>
<th>Quality Model</th>
<th>External and Internal Quality</th>
<th>Functional</th>
<th>Reliability</th>
<th>Efficiency</th>
<th>Maintainability</th>
<th>Testability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suitability</td>
<td>Reliability</td>
<td>Function</td>
<td>Maintainab</td>
<td>Testable</td>
<td>Changeable</td>
<td>Installable</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Interoperability</td>
<td>Function</td>
<td>Maintainab</td>
<td>Testable</td>
<td>Changeable</td>
<td>Installable</td>
</tr>
<tr>
<td>Interoperability</td>
<td>Functionality</td>
<td>Function</td>
<td>Maintainab</td>
<td>Testable</td>
<td>Changeable</td>
<td>Installable</td>
</tr>
<tr>
<td>Security</td>
<td>Functionality</td>
<td>Function</td>
<td>Maintainab</td>
<td>Testable</td>
<td>Changeable</td>
<td>Installable</td>
</tr>
<tr>
<td>Functionality</td>
<td>Functionality</td>
<td>Function</td>
<td>Maintainab</td>
<td>Testable</td>
<td>Changeable</td>
<td>Installable</td>
</tr>
<tr>
<td>Usability</td>
<td>Functionality</td>
<td>Function</td>
<td>Maintainab</td>
<td>Testable</td>
<td>Changeable</td>
<td>Installable</td>
</tr>
<tr>
<td>Compliance</td>
<td>Functionality</td>
<td>Function</td>
<td>Maintainab</td>
<td>Testable</td>
<td>Changeable</td>
<td>Installable</td>
</tr>
</tbody>
</table>
Analytical Product Quality Assurance:
Static testing: test metrics (9/11)

<table>
<thead>
<tr>
<th>Test Specification</th>
<th>Quality Assurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Effectiveness</td>
<td>Reliability</td>
</tr>
<tr>
<td>Stability</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Usability</td>
<td>Usability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Usability</td>
</tr>
<tr>
<td>Usability</td>
<td>Usability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Usability</td>
</tr>
<tr>
<td>Usability</td>
<td>Usability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Usability</td>
</tr>
<tr>
<td>Usability</td>
<td>Usability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Usability</td>
</tr>
<tr>
<td>Usability</td>
<td>Usability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Usability</td>
</tr>
<tr>
<td>Usability</td>
<td>Usability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Usability</td>
</tr>
<tr>
<td>Usability</td>
<td>Usability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Usability</td>
</tr>
<tr>
<td>Usability</td>
<td>Usability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Usability</td>
</tr>
</tbody>
</table>

Quality Assessment:
Static testing: test metrics (10/11)

- How do we define metrics for the quality model?
- Goal, Question, Metric (GQM) approach from Basili and Weiss (1984)
  1. State the goal to be achieved due to the measurement.
  2. Develop questions that breaks the goal into its major components. What are the attributes that need to be captured?
  3. Develop metrics that answer the questions, preferably free from side-effects.

Example for GQM:
- **Goal:** Evaluate the changeability of templates in a TTCN-3 specification
- **Questions:**
  - Are inline templates used?
  - Are there many duplicate inline templates?
  - What is the degree of test data duplication?
- **Metrics:**
  - Number of inline templates
  - Number of inline templates that are duplicates
  - Number of templates with duplications
  - Number of template definitions

Analytical Product Quality Assurance:
Static testing: smell detection (1/2)

- Patterns of possibly inappropriate code usage.
- Notion of metrics and code smells not disjoint:
  - Code smell → Metric: count occurrences of code smell.
  - Metric → Code smell metric violates boundary.
- > 38 TTCN-3 code smells identified (master’s thesis of Martin Bisanz).
- Automatic detection of test smells using static analysis:
  - Detection of patterns in the Abstract Syntax Tree (AST)
  - Violation of metric thresholds.

Analytical Product Quality Assurance:
Static testing: smell detection (2/2)

<table>
<thead>
<tr>
<th>Metric</th>
<th>TTCN-3 Code Smell</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP</td>
<td>42376</td>
</tr>
<tr>
<td>IPa</td>
<td>40503</td>
</tr>
<tr>
<td>Lines of code</td>
<td>528</td>
</tr>
<tr>
<td>Number of testcases</td>
<td>317</td>
</tr>
<tr>
<td>Number of components</td>
<td>100</td>
</tr>
<tr>
<td>Duplicate at Branch</td>
<td>253</td>
</tr>
<tr>
<td>Activation Asymmetry</td>
<td>78</td>
</tr>
<tr>
<td>Distinct Magic Values</td>
<td>115</td>
</tr>
<tr>
<td>Unused Definitions</td>
<td>50</td>
</tr>
<tr>
<td>195</td>
<td></td>
</tr>
</tbody>
</table>
Analytical Product Quality Assurance

Static Testing

With Computer

Guideline Checking
Test Metrics
Small Detection

Code Review

Improvement: test refactoring

Analytical Test Quality Assurance

Dynamic Testing

Simulation Model-Based Analysis

Analytical Test Quality Assurance

Analytical Product Quality Assurance: Dynamic testing: model-based analysis (1/6)

- Testing test specifications?
  - Unreasonable!
  - We would need tests for the tests that test the tests etc.

- Our solution:
  - Simulation of test behavior (branch coverage)
  - Construction of a model from the simulated test runs
  - Checking the model for anomalies using temporal logic
  - Result can essentially be considered an testing of specific properties

Analytical Product Quality Assurance: Dynamic testing: model-based analysis (2/6)

- Structural Property
- Linear Temporal Logic
- Spin / Promela Processes
- Model Checking with Spin

Analytical Product Quality Assurance: Dynamic testing: model-based analysis (3/6)

- No verdict set!

Analytical Product Quality Assurance: Dynamic testing: model-based analysis (4/6)

- Simple possible LTL formula:
  - For each path (no verdict is set) until a verdict becomes either pass, inconclusive or fail
  - $P \land \neg\text{incon}\land \text{fail} \rightarrow \neg\text{pass} \lor \text{incon} \lor \text{fail}$
  - More sophisticated and comprehensive formulas possible

Analytical Product Quality Assurance: Dynamic testing: model-based analysis (5/6)

- Result:
  - The structural property does not hold in all possible paths.
  - The failing traces.

Test Component Models

Model Checking with Spin
Analytical Product Quality Assurance:
Dynamic testing: model-based analysis (6/6)

- Further possibilities:
  - Deadlocks
  - Idle PTC
  - Default Asymmetries
  - ...
- 10+ properties documented (work in progress)

Analytical Product Quality Assurance:
Improvement: test refactoring (1/6)


- Refactoring is a systematic way to restructure test code – no bugfixing, no new functionality
- Improves quality characteristics like:
  - Usability
  - Maintainability
  - Reusability
- TTCN-3 refactoring catalogue with more than 50 refactorings:
  - Test behavior, test data, overall test suite structure
  - http://www.trex.informatik.uni-goettingen.de/trac/wiki/TTCN3RefactoringCatalogue

Analytical Product Quality Assurance:
Improvement: test refactoring (3/6)

Refactoring 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.

Analytical Product Quality Assurance:
Improvement: test refactoring (4/6)

TTCN-3 example (unrefactored):

```java
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 ("127.0.0.2"));
}
}"
```
Analytical Product Quality Assurance: Improvement: test refactoring (5/6)

- TTCN-3 example (refactored):

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

Analytical Product Quality Assurance: Improvement: test refactoring (6/6)

- Effects on the number of templates (experiments):

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETSI IP 4.1.1</td>
<td>11.5%</td>
</tr>
<tr>
<td>ETSI IP 6.1.1</td>
<td>53.4%</td>
</tr>
<tr>
<td>ETSI IP 8.2.1</td>
<td>36.8%</td>
</tr>
</tbody>
</table>

Tools for general-purpose languages

- Guideline checking and smell detection:
  - Java: Findbugs, PMD, Checkstyle, Sonar (free)
  - C/C++: Coverity Prevent, PCLint (commercial)
  - C#: FxCop, StyleCop, devAdvantage

- Software metrics:
  - Java: Eclipse Metrics Plugin (free), CodePro AnalytiX (commercial)
  - C#: DevMetrics (free)

Tools developed in Göttingen

- TRex
  - Metrics
  - Dynamic Analysis
  - Refactoring
- ETSICheck
  - TTCN-3 Guideline Checker
- ETSIDOC
  - TTCN-3 Documentation Generator (HTML/PDF)
- Tool for Model-Based Analysis
  - Reverse-Engineering Models from TTCN-3 Specifications
  - Analyzing Properties using Model-Checking

Agenda

- Organizational Quality Assurance
- Constructive Product Quality Assurance
- Analytical Product Quality Assurance
- Tools
- Summary / Conclusion
Summary

- Today’s test specifications are complex
  - Quality Assurance (QA) is necessary

- QA aspects to be considered:
  - Organizational QA
    - Test process model, test process improvement model
  - Constructive Product QA
    - Training, code / style guidelines, test patterns, …
  - Analytical Product QA
    - Static testing, dynamic testing, refactoring, debugging/correction

Conclusion

- QA in testing involves more than coding guidelines!
- QA in testing needs to be considered early on
  - Example: consideration of tool support in decision making
- This talk covered only selected aspects of test QA
  - Organizational aspects were only indicated
  - Test process metrics not covered
  - Test debugging methodologies not covered
- Raise the awareness for QA in testing!

Thank you for your attention!

Literature (1/6)

- Organizational Quality Assurance:

- TTCN-3 Tools:
  - http://www.trex.informatik.uni-goettingen.de

Literature (2/6)

- General-Purpose Tools:
  - http://www.eclipse.org
  - http://www.netbeans.org
  - http://metrics.sourceforge.net
  - http://www.instantiations.com/codepro
  - http://www.slickedit.com
  - http://valgrind.org
  - http://sonar.codehaus.org
  - http://coverity.com
  - http://www.gimpel.com
  - http://code.msdn.microsoft.com/sourceanalysis
  - http://www.anticipatingminds.com
Literature (3/6)

- **Constructive Quality Assurance:**

Literature (4/6)

- **Analytical Quality Assurance:**
  - Static Testing of Test Specifications:

Contact

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

- **E-Mail:**
  - grabowski@cs.uni-goettingen.de
  - zeiss@cs.uni-goettingen.de

Special Thanks

- Andreas Ulrich, Siemens AG
- Philip Makedonski, University of Göttingen