Towards Quality of TTCN-3 tests

Diana Vega, George Din, Cosmin Rentea

Contents

- Motivation
- Towards quality assessment of tests
- TTCN-3 Overview
- Definition of the quality of a test suite
- Metrics definition for TTCN-3
- Design for a TTCN-3 metrics tool
- Conclusions and Outlook
Motivation

- Measurement is critical in **software quality assurance**
  - software metrics were introduced
  - product, process, project metrics - estimation, resource planning and scheduling
  - quantitative managerial decision-making during the software life-cycle
  - quality models, intern metrics, external metrics, etc. (ISO 9126)

- Testing: software quality measurement
  - decisive in software quality assessment

Quality

- Quality estimation of software product
  - product quality is commonly recognized as lack of "bugs" in the product
    - defect rate (e.g., number of defects per million lines of source code)
    - reliability (e.g., mean time to failure, probability of failure-free operation in a specified time)
  - quality assessment - part of software development process

- Quality estimation of testing
  - how good the tests are?
  - research oriented theme
  - structural and functional testing aspects
  - TTCN-3 testing language – a good test environment for black-box testing quality analysis and estimation

Associated Concepts:
- SUT (System Under Test)
- test suite
- test cases
- test stimuli & oracle
- test system
Quality of Structural Testing

- System under test (SUT) is transparent to the Test System
  - glass-box, white-box

- Coverage - measure of test completeness
  - Code coverage analyzer aims to uncover software faults
  - coverage measures:
    - statement coverage
    - branch coverage
    - line coverage

Quality of Functional Testing

- Program behavior is compared against the SUT specification
  - black-box
  - internals of the SUT cannot be seen

- Model of the specification
  - state machines: FSM, ASM, EFSM
  - Markov chains
  - other formal specification techniques

- Specification-based coverage - measurement for the completeness of testing
  - state coverage
  - transition coverage
  - transition sequences coverage
Methods Used in Functional Testing

- **Research**: reduce black-box testing efforts while maintaining the quality of the testing
  - Input-Output (IO) Analysis
  - combinatorial testing techniques
  - various combination and heuristic strategies
    - Category Partition (CP)
    - Equivalence Partitioning (EP)
    - Artificial Intelligence (test data generation)

TTCN-3 Overview

- **Abstract test specification (ATS)**
  - data templates allow unlimited structuring and reusability of test data
  - matching mechanism to compare an oracle to response data
  - communication paradigms: message and procedure oriented ports
  - parallel test components

- **Concrete test implementation**
  - adapter and codec (coder/decoder of data types)

- General architecture used for black-box distributed testing, standardized by ETSI
- High-level language for abstract specification of test logic
- TTCN-3 is used for
  - all types of reactive systems
- Execution environment
  - TRI, TCI
- Multiple presentation formats
  - text, graphical, tabular
Quality Definition of a Test Suite

- Always associated to a **criterion**
  - E.g.: reusability of test data types, test stimuli that cover all system input domains, etc.

- Quality is a function
  - $f(S, I, T, C)$
    - $S$ - the SUT Specification
    - $I$ - the Implementation of the Specification $S$
    - $T$ - the Test Suite whose quality is to be assessed
    - $C$ - Criterion upon which the quality is evaluated

Aspects of Test Suite Effectiveness

- Depending on the **selected criterion** ($C$):
  - (1) Relative manner
    - versions of test suite $T$ ($T_1$, $T_2$, ..., $T_n$) a specific quality of SUT is observed
    - close idea: mutant testing
    - Example: $C$ – revealed failures

    ![Test Suite Diagram]

  - (2) Absolute manner
    - Quality of a test - one percentage level between an established lower and a threshold value
    - Example: $C$ - usage of all defined variables
Effectiveness of a TTCN-3 Test Suite

- Two points of view:
  - (1) Abstract Test Suite (ATS)
    - SUT interface representation is exhibited to TTCN-3 test infrastructure by means of the TSI (test system interface), test data type descriptions, etc.
  - (2) Executable Test Suite (ETS)
    - communication with SUT

- Analysis of test suite effectiveness
  - Statically
    - inspecting the ATS without running the test against SUT
    - TTCN-3 static metrics
  - Dynamically
    - verdicts establishment
    - execution artifacts (e.g. time)
    - TTCN-3 dynamic metrics

General TTCN-3 Static Metrics

- Static metrics similar to other programming languages
  - lines of code: the number of all lines of code contained in a TTCN-3 test suite; commented lines of code
  - number of words: number of words contained in a test suite obtained by summing up the words contained in each module
  - test suite size: number of bytes for a test suite obtained by summing the size of each module
  - min, max, average lines of code for a function
  - min, max, average number of parameters for a function
  - number of unused variables: counted with respect to their closer visibility domain (scope) e.g. module, function, test case, etc.
  - min, max, average cyclomatic complexity for a function: number of branches in the function
  - min, max, average function Fan-in for a test suite: how many entities call the function
  - min, max, average function Fan-out for a test suite: how many entities are called from the function
Specific TTCN-3 Static Metrics

- Static metrics TTCN-3 specific
  - min, max, average lines of code for a test case
  - min, max, average altstep Fan-in for a test suite: how many entities call the altstep
  - min, max, average altstep Fan-out for a test suite: how many entities are called from the altstep
  - min, max, average number of parameters for a test case
  - max size of modules dependency: the maximum number of imported modules contained in a module from the test suite (non-recursive computation)
  - max depth of modules dependency: the max depth level of the nested modules (recursively computed depth of imported modules related to a module)
  - min, max depth of a user-type definition
  - max spreading of user-defined type: maximum number of fields of a TTCN-3 user-defined type
  - max depth of a type definition of a template sent via a port
  - max depth of inheritance of templates by means of modifiers
  - min, max, average number of parameters for a template
  - min, max, average number of ports for a component
  - min, max, average test case call for execution depth in a test suite: the nesting level of calling for execution in control part of a test case

TTCN-3 Dynamic Metrics

- Decisive factor in analyzing the quality of a test
- Based on the information recorded during or after test execution
  - number of passed/failed test cases after test suite execution
  - min, max, average time for the execution of a test case in a test suite
  - min, max, average number of parallel test components (PTCs) employed for the execution of a test case
- Defined for specific type of applications
  - e.g. inter-failure time computation (real-time systems)
- Time evolution perspective
  - statistics related to test cases that changed their verdicts
**Metrics collector Tool**

- **TTCN-3 static metrics collector**
  - Requirements: TTCN-3 ATS specification environment
  - Solution:
    - TTworkbench product selection
      » Eclipse-based test platform
    - TTCN-3 metamodel stored in a EMF repository
    - TTCN-3 metrics plug-in developed upon TTworkbench infrastructure
      - work-in-progress

- **TTCN-3 dynamic metrics analyzer**
  - tool built into TTCN-3 Runtime and Logging interfaces
  - off-line investigation of a very detailed log file produced after test execution

---

**IMS Static Metrics**

<table>
<thead>
<tr>
<th>Static metrics</th>
<th>ATS v.1</th>
<th>ATS v.2</th>
<th>ATS v.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of modules</td>
<td>5</td>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>Size (bytes)</td>
<td>161464</td>
<td>187240</td>
<td>195132</td>
</tr>
<tr>
<td>Words</td>
<td>16381</td>
<td>19324</td>
<td>21391</td>
</tr>
<tr>
<td>Lines of Code</td>
<td>5801</td>
<td>6762</td>
<td>7002</td>
</tr>
<tr>
<td>Number of Functions</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Min Lines of Code for a Function</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Max Lines of Code for a Function</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Average Lines of Code for a Function</td>
<td>3.5</td>
<td>4</td>
<td>2.83</td>
</tr>
<tr>
<td>Number of Test Cases</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Min Lines of Code for a Test Case</td>
<td>91</td>
<td>32</td>
<td>48</td>
</tr>
<tr>
<td>Max Lines of Code for a Test Case</td>
<td>111</td>
<td>120</td>
<td>158</td>
</tr>
<tr>
<td>Average Lines of Code for a Test Case</td>
<td>101</td>
<td>82.5</td>
<td>119.13</td>
</tr>
<tr>
<td>Max Size of Modules Dependency</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Min number of parameters for a function</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max number of parameters for a function</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Average number of parameters for a function</td>
<td>1</td>
<td>1</td>
<td>0.89</td>
</tr>
<tr>
<td>Min number of parameters for a test case</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max number of parameters for a test case</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Average number of parameters for a test case</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of Component Types</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Min number of ports for a component</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Max number of ports for a component</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Conclusion & Outlook

- **Static and dynamic metrics** for TTCN-3 were introduced
  - targeting the analysis and estimation of the test's quality

- Design ideas for a TTCN-3 metrics tool implementation

- Find a correlation between statically and dynamically collected values
  - statistical theory basis
  - in which measure the combination between the proposed metrics better assess the testing quality