Modelling und Testing: Model-Based Testing

Ina Schieferdecker
ina.schieferdecker@fokus.fraunhofer.de
+49 30 3463 7241

Outline

- Motivation
- Principles of Modell-Based Testing
- An Example Process
- Summary
Current Situation for Software-Based Systems

Number of successful software projects is decreasing

Criticality of an error depends on the phase in which it is created

Error correction costs increase dramatically the later errors are detected

Design and testing take the majority of system development

*source: survey of one of our main customers

System and Test system

Although programmers, testers and program managers know that code must be designed and tested, many appear to be unaware that tests themselves must be designed and tested – by a process no less rigorous and no less controlled than that used for code.

Boris Beizer
The W-Model

- Reflected in the V-Modell XT by check lists
- No explicit requirement for such checks – a design issue
  - Model-based testing as a way forward

Model-Based Testing

- ... refers to software testing where test cases are derived in whole or in part from a model that describes selected, often structural, functional, sometimes non-functional aspects of a system under test (SUT).
  - [Wikipedia, 2007]

...
Model-Based Testing (2nd)

- Various system models exist such as:
  - Requirements models
  - Information models
  - Work flow models
  - Architectural models
  - Behavioural models
  - Configuration models
  - Deployment models
  - Performance models
  - Risk models
  - …

- hence, model-based test methods differ in the system model being considered, the methods taken for test generation and the way test results are being obtained

MBT Placement in Testing

Test Methods

- static
  - manual
  - automated
    - rule based
    - model based
  - systematic
    - active
    - passive
    - model based

- dynamic
  - informal
    - code based
    - model based
    - rule based
    - model based
Principal Approach

System model represents Test model realizes
System represents Test system realizes
System represents Test model realizes
requirements validates validates realizes

Different Forms

System-driven

(a) System model generates Test model
System \(\xrightarrow{\text{test}}\) Test system

Test-driven

(b) Test model generates System Test system
System \(\xrightarrow{\text{test}}\) Test system

(c) System Test model generates System Test system
Different Forms (2\textsuperscript{nd})

\textit{Combined models}

\textbf{(d)}

\begin{center}
\begin{tikzpicture}[node distance=2cm, auto]
  \node (system) [circle,draw] {System model};
  \node (test) [circle,draw, below of=system, xshift=2cm] {Test model};
  \node (test_system) [circle,draw, below of=test] {test system};
  \path[->] (system) edge node {generate} (test); 
  \path[->] (test) edge node {generate} (test_system); 
\end{tikzpicture}
\end{center}

\textbf{(e)}

\begin{center}
\begin{tikzpicture}[node distance=2cm, auto]
  \node (system) [circle,draw] {System model};
  \node (test) [circle,draw, below of=system, xshift=2cm] {Test model};
  \node (test_system) [circle,draw, below of=test] {test system};
  \path[->] (system) edge node {generate} (test); 
  \path[->] (test) edge node {generate} (test_system); 
\end{tikzpicture}
\end{center}

Different Forms (3\textsuperscript{rd})

\textit{Combined, but independent models}

\textbf{(f)}

\begin{center}
\begin{tikzpicture}[node distance=2cm, auto]
  \node (system) [circle,draw] {System model};
  \node (test) [circle,draw, below of=system, xshift=2cm] {Test model};
  \node (test_system) [circle,draw, below of=test] {test system};
  \node (requirements) [rectangle, above of=system, xshift=2cm] {Requirements};
  \path[->] (system) edge node {generate} (requirements); 
  \path[->] (test) edge node {generate} (requirements); 
  \path[->] (requirements) edge node {derive} (system); 
  \path[->] (requirements) edge node {derive} (test); 
\end{tikzpicture}
\end{center}
## UML-Based Test Generation

<table>
<thead>
<tr>
<th>UML diagram</th>
<th>Test Level</th>
<th>Test Kind</th>
<th>Test Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unit</td>
<td>Component</td>
<td>Integration</td>
</tr>
<tr>
<td>Structure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>composite structure</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>component</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>deployment</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>object</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>package</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behavior</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>use case</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>activity</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>sequence</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>communication</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>interaction overview</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>timing</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>state machine</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

1 By use of additional profiles

---

## Test Specification Techniques

- **The Testing and Test Control Notation**
  - A standardized alternative to proprietary test systems
    - Developed by a large group of testing experts
    - Used by a growing community
    - Proven by tools
    - Maintained at ETSI

- **A test specification and implementation language**
  - A multipart standard covering
    - textual TTCN-3 core
    - graphical TTCN-3
    - execution interfaces TRI and TCI
    - language mappings to TTCN-3, e.g. for IDL
Test Specification Techniques (2nd)

- TTCN-3: one test technology for different tests
  - Distributed, platform-independent testing
  - Integrated graphical test development, documentation and analysis
  - Adaptable, open test environment

- The testing middleware
  - unifying the documentation and definition of tests
  - unifying the tests in IT, Internet- and Telco-based systems (supporting their convergence)
  - unifying the test infrastructure

- Areas of Testing
  - Regression Testing
  - Conformance/Functionality Testing
  - Interoperability/Integration Testing
  - Load/ Stress Testing

---

Test Specification Techniques (3rd)

- Test design with the UML Testing Profile
  - Integrated into UML based development processes

- U2TP has been defined to capture all information that would be needed by different test processes

- It is a testing profile based upon UML 2.0
  - That enables the test definition and test generation based on structural (static) and behavioral (dynamic) aspects of UML models, and
  - That is capable of inter-operation with existing test technologies for black-box testing

- It is an official OMG standard since summer 2005
### Test Specification Techniques (4th)

<table>
<thead>
<tr>
<th></th>
<th>U2TP</th>
<th>TTCN-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test design</td>
<td>✔️</td>
<td>(—you)</td>
</tr>
<tr>
<td>Test specification</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Test execution</td>
<td>(—you)</td>
<td>✔️</td>
</tr>
<tr>
<td>Format</td>
<td>Graphical</td>
<td>Textual and graphical</td>
</tr>
<tr>
<td>Transformation</td>
<td>U2TP to TTCN-3 (✔️)</td>
<td>TTCN-3 to U2TP ✔️</td>
</tr>
</tbody>
</table>

---

### The Overall Picture

The overall picture of the modell-based test process is depicted below:

1. Test planning
2. Risk assessment
3. Test generation
4. Test validation
5. Test realization
6. Test execution
7. Test selection
8. Resource planning
9. Test evaluation

The process starts with the system model, which is the basis for all other activities. Each step leads to the next, culminating in the executable test suite.
OMG MDA: A Model-Centric System Development

MDA Extended: Integrated Modelling and Testing
A Small Example

d. ...

e. Derive test configurations

f. Derive interfaces

g. Derive test behavior

h. ...

Run the Tests and Evaluate the Test Results
Summary

- Basic idea of model based testing approach:
  Testing complex software by means of
  exhaustive coverage of relatively simple models
- Area of applicability:
  Any software and hardware components with
  well-defined interfaces or functional properties
- Formalization of requirements allows
  improving most phases of software development processes
- Formalized methods should be
  well integrated in existence tools and processes
- Tools:
  - Numerous TTCN-3 tools, some U2TP tools
  - Selected test generation tools

MBT means also Test Models: TTCN-3 and U2TP

- Both are
  - test specification techniques
  - open to various domains
  - open to various development and test processes
- TTCN-3 is widely supported by tools, U2TP is emerging
- U2TP enables test design along UML based design processes
- TTCN-3 is the choice for technical testing and automated execution
- U2TP is the choice for high-level test design and modelling
- Both are a basis for systematic testing and can be integrated into the system development
Improvement Potentials

- Well-defined test processes in place
- Emphasis on integration and system tests
- Test development and specification like in system development
- Test development as early as possible and as continuous as possible
- Test automation in order to concentrate on the what to be tested and not the how

- Test generation
  - Testability of models
  - Test hypotheses, heuristics for efficient and effective test generation
  - Adequate reflection of test data in test generation
  - Definition and use of test patterns

- Test quality measures
- Traceability from requirements via models to tests, test campaigns and system bugs
- ...

Thank you!

Any further questions?