Cut your software testing expenses

Model based testing and TTCN-3 explained on a case study

NORDIC TESTING DAYS 2012
4-5 June, Tallinn, Estonia

andrus.lehtmets@elvior.com
Elvior in a nutshell

► Elvior was founded in 1992 and started out being a subcontractor for top-tier telecommunication companies in Europe

► Developed from being subcontractor into an independent testing tool and services provider

► Extensively focused on TTCN-3 testing and model-based black-box testing of software intense systems

► Main business line is test automation (tools and services)
Cut your software testing expenses

**Agenda**

1. Overview of model-based testing (MBT)
2. Tools used in practical exercise
3. Example SUT (Light switch)
4. Workflow of MBT
5. System Adapter (SA) used in example
6. Light switch state model
7. Test cases generation
8. Execution of generated TTCN-3 test cases
9. Example of real industrial case study
10. Questions
Model Based Testing (MBT) – what is it?

► is software testing where
► from a **model** that describes some (usually functional) aspects of the system under test
► + **model coverage criterion**
► **test cases (scripts)** are derived automatically by some tool
Cut your software testing expenses

**TTCN-3 – what is it?**

- Testing and Test Control Notation Version 3
  - TTCN-3 Testing Language
    - Standardized modern high level language for testing
  - TTCN-3 Architecture
    - TTCN-3 TCI and TRI interfaces
- Black-box, functional testing of reactive systems
Cut your software testing expenses

**TTCN-3 testing**

- SUT specification
- Test goal
- Test code/script TTCN-3
- System Under Test (SUT)
- TTCN-3 Test Tool
Cut your software testing expenses

Model Based Testing and TTCN-3

- **SUT specification**
- **SUT state model**
- **Test goal**
- **Test Generator**
- **Test code/script TTCN-3**
- **TTCN-3 Test Tool**

**System Under Test (SUT)**
MBT – when to use?

► it is possible to formalize system behavior OK / not NOK
► functional testing OK / GUI testing NOK
► automated testing OK / manual testing NOK
► it must be possible to control testing by test script and SUT behavior must be observable

Test tool
Test script

System Under Test (SUT)

observable events

controllable events
Benefits of MBT

► Writing and maintenance of test scripts is a time and effort consuming task.

► **Better tests.** Easier and cheaper to generate sufficient amount of test scripts to achieve a good enough test coverage.

► **Lower costs.** Work effort for test suite maintenance will reduce significantly.
  
  ► Instead of maintaining huge amount of test scripts the test engineer should maintain a SUT model only.
  
  ► If there are changes in the behaviour of the SUT then it is rather easy to update the model correspondingly and re-generate all test scripts once again.
Cut your software testing expenses

Classical expectations to MBT

1. Through formalization discloses ambiguity in specifications and helps validation of specifications.
2. Better test coverage.
3. Cost effective in maintenance phase.
Cut your software testing expenses

**Different approaches** *(model building and tests generation)*

- Building SUT state model
  - Drawing with some UML tool *(Poseidon, Artisan, TestCast MBT modeler)*
  - Using some model building language *(nModel)*

- Generated tests - executable at once
  - Real test data *(to and from SUT)* and
  - TTCN-3 configuration is used when building the model

- Generated tests use special adapter *(TTCN-3)* for execution
  - SUT model is on higher abstraction level
  - Send and receive templates to and from SUT are filled by the *(TTCN-3)* special adapter
Cut your software testing expenses

Test environment (Elvior approach)
Cut your software testing expenses

Tools – TestCast MBT Designer (powered by Conformiq)

► Used for creating SUT model
► Java like action language (QML)
► Graphical modeler
Cut your software testing expenses

Tools – TestCast MBT Designer (powered by Conformiq)

- Used for generating tests (TTCN-3 scripts)
  - uses SUT model and selected test coverage
  - runs on Eclipse platform
Cut your software testing expenses

Tools – **TestCast Professional** *(Elvior TTCN-3 test tool)*

- Used for development and execution of TTCN-3 tests
  - uses test scripts generated by TestCast MBT Generator
  - supported operating systems: Windows and Linux
Cut your software testing expenses

SUT–LightSwitch—the example SUT

The system under test (SUT) is a lighting system that consists of a switch that turns lights on or off at the user’s request.
SUT–LightSwitch—the example SUT (requirements)

- The light shall be **switched on** by the request from the controlling environment,
- The light shall be **switched off** by the request from the controlling environment.
- If the light is already on/off, requesting the same operation (turning light on/off respectively) shall not change the system state.
- If SUT receives not supported command, then it notifies the controlling environment.
# Cut your software testing expenses

## SUT—LightSwitch—the example SUT (use cases)

<table>
<thead>
<tr>
<th>#</th>
<th>Precondition</th>
<th>Input (to the SUT)</th>
<th>Expected result (from the SUT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Light is off</td>
<td>Command turnOn</td>
<td>lightIsOn</td>
</tr>
<tr>
<td>2</td>
<td>Light is on</td>
<td>Command turnOff</td>
<td>lightIsOff</td>
</tr>
<tr>
<td>3</td>
<td>Light is off</td>
<td>Command turnOff</td>
<td>lightIsOff</td>
</tr>
<tr>
<td>4</td>
<td>Light is on</td>
<td>Command turnOn</td>
<td>lightIsOn</td>
</tr>
<tr>
<td>5</td>
<td>Light is on or off</td>
<td>Unknown command</td>
<td>Unrecognised command</td>
</tr>
</tbody>
</table>
Cut your software testing expenses

**SUT—LightSwitch—the example SUT** (interface)

1. SUT interacts with outside world using console interface (standard input/output)
2. iLights interface defines commands and SUT responses

<table>
<thead>
<tr>
<th>#</th>
<th>Input (to the SUT)</th>
<th>Output (from the SUT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>string command</td>
<td>string currentLampState</td>
</tr>
</tbody>
</table>

Text constants for input and respective output

<table>
<thead>
<tr>
<th>#</th>
<th>Input (to the SUT)</th>
<th>Output (from the SUT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>ready</td>
</tr>
<tr>
<td>2</td>
<td>turnOn</td>
<td>lightIsOn</td>
</tr>
<tr>
<td>3</td>
<td>turnOff</td>
<td>lightIsOff</td>
</tr>
<tr>
<td>4</td>
<td>xyz</td>
<td>Unrecognized command</td>
</tr>
<tr>
<td>5</td>
<td>exit</td>
<td></td>
</tr>
</tbody>
</table>
Cut your software testing expenses

**SUT—LightSwitch—the example SUT** *(T3 test environment)*

Adapter between test tool and SUT is needed.
Cut your software testing expenses

**System Adapter used in example**

- System Adapter (SA) connects testing tool (TestCast (TC)) with System Under Test (SUT).
- TRI - TTCN-3 standardizes interface between testing tool and SA, this interface is called TTCN-3 Runtime Interface.
- Interface between SA and SUT is always proprietary and therefore needs to be implemented within SA.

- TRI interface is mapped for different languages (C, C++, C#, Java) (Part 5: TTCN-3 Runtime Interface)
- Implementation is tool dependent.
- Most important is what to implement in the methods of the interfaces (i.e. triSend, triEnqueueMsg, triMap)
Cut your software testing expenses

Workflow of MBT (Elvior approach)

► Create SUT model.
► Prepare test data, configuration, functions etc.
► Create system adapter according to TTCN-3 TRI.
► Create codecs.
► Generate tests for specified test goal.
► Tune TTCN-3 adapter
► Execute tests.
► Evaluate results and continue with next increment.
Cut your software testing expenses

State Model of SUT

<table>
<thead>
<tr>
<th>Starting state</th>
<th>Trigger (cmd)</th>
<th>Effect (output)</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>LightSwitch_Off</td>
<td>TurnOn</td>
<td>LightIsOn</td>
<td>LightSwitch_On</td>
</tr>
<tr>
<td>LightSwitch_On</td>
<td>TurnOff</td>
<td>LightIsOff</td>
<td>LightSwitch_Off</td>
</tr>
<tr>
<td>LightSwitch_Off</td>
<td>TurnOff</td>
<td>LightIsOff</td>
<td>LightSwitch_Off</td>
</tr>
<tr>
<td>LightSwitch_On</td>
<td>TurnOn</td>
<td>LightIsOn</td>
<td>LightSwitch_On</td>
</tr>
<tr>
<td>LightSwitch_On</td>
<td>UnknownCmd</td>
<td>LightIsOn</td>
<td>LightSwitch_On</td>
</tr>
<tr>
<td>LightSwitch_Off</td>
<td>UnknownCmd</td>
<td>LightIsOff</td>
<td>LightSwitch_Off</td>
</tr>
</tbody>
</table>
Cut your software testing expenses

Model in QML action language

```csharp
/** Declaration of the external interface of the system block: a similar construct to programming languages. In this "system block", the interface (in) and one outbound interface (out) are the names for the interfaces in the list the types of records that can possibly go question. */

system
{
  Inbound ilights : Command;
  Outbound ilights_out : Response;
}

/** Declaration of a message type, which is technologically specific. It is a record of pure data. This record i.e. it does not contain any actual data field.

record Command {
  String cmd;
}

record Response {
  String rsp;
}

class LightSwitch extendsStateMachine
{
  /** The default constructor. */
  public LightSwitch() {
  }

  private void ready() {
    Response r;
    rrsp= "lamp implementation version 1.0 res ilights_out.send(r)";
  }

  private void lightIsOn() {
    Response r;
    rrsp= "lightIsOn";
    ilights_out.send(r);
  }

  private void lightIsOff() {
    Response r;
    rrsp= "lightIsOff";
    ilights_out.send(r);
  }

  private void unrecognizedText() {
    Response r;
  }
}*/
```
Cut your software testing expenses

Test cases generation
Cut your software testing expenses

**Execution of generated test cases**

Precondition: system adapter exists (TRI), SUT is reachable.
Cut your software testing expenses

Analyzing test execution results
Industrial case study - Feeder Box Control Unit

Feeder Box Control Unit (FBCU). It is a subsystem of the street lighting control system functioning today in Tartu, the second biggest city of Estonia.
Cut your software testing expenses

Industrial case study – test environment

Poseidon → XMI → TestCast Generator

TTCN-3

TestCast Professional - TTCN-3 test tool

TRI → System Adapter

TestCast-LabView adapter

Messages over TCP/IP

LabVIEW

USB

Power supply module

Digital/analog module

USB

Hardware adapter

FBCU (SUT, hybrid embedded system)
Cut your software testing expenses

Industrial case study – SUT state model

Model of FBCU power management (31 states, 73 transitions)
Using MBT in this case study is very efficient, because FBCU behavior is complex and it is easier to change model than rewrite test code – proved in practice.

Numbers (first increment):

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Code lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TTCN-3 code (messages, test data, configuration)</td>
<td>15 days</td>
</tr>
<tr>
<td>2</td>
<td>System adapter</td>
<td>150 days</td>
</tr>
<tr>
<td>3</td>
<td>Model building</td>
<td>45 days</td>
</tr>
<tr>
<td>4</td>
<td>Generated tests</td>
<td>NA</td>
</tr>
</tbody>
</table>
Cut your software testing expenses

Industrial case study – results, increment 2

FBCU changed significantly, new model was built from scratch.

Numbers (second increment):

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Code lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>~ 10 days</td>
<td>NA</td>
</tr>
<tr>
<td>4</td>
<td>NA</td>
<td>~ 20 000</td>
</tr>
</tbody>
</table>

3 fatal bugs found.
Conclusion

- There are common tasks to be solved in both cases (manual and model based TTCN-3 testing).
- Using MBT with TTCN-3 gives extra advantage (TTCN-3 is dedicated for tests, it is natural to generate TTCN-3).
- Building the model formalizes SUT behavior and therefore discloses ambiguity in SUT specifications.
- Model building is resources consuming work, it pays back in maintenance phase – it is easier to alter model and generate tests again.
- MBT advantages are more visible with complex SUT models.
- MBT gives very handy approach for exploratory testing.
Cut your software testing expenses

September 25-27, 2012

Cut your software testing expenses

Thank you!

Questions?

andrus.lehtmets@elvior.com

www.elvior.com