Model-Based Testing

An Evaluation

Johan Nordholm
johan.nordholm24@gmail.com
# Background and Motivation

## Traditional testing processes
- Common factors:
  - Manual test design & Manual test coverage analysis

## Model-based testing automates these

<table>
<thead>
<tr>
<th>Testing Process</th>
<th>Test cases</th>
<th>Test execution</th>
<th>Test coverage</th>
<th>Test result analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture/Replay</td>
<td>Manual design</td>
<td>Automated execution (records manual execution)</td>
<td>Manual analysis</td>
<td>Automated analysis (manually written)</td>
</tr>
<tr>
<td>Script-based</td>
<td>Manual design</td>
<td>Automated execution</td>
<td>Manual analysis</td>
<td>Automated analysis (manually written)</td>
</tr>
<tr>
<td>Keyword-based</td>
<td>Manual design</td>
<td>Automated execution</td>
<td>Manual analysis</td>
<td>Automated analysis (manually written)</td>
</tr>
<tr>
<td>Model-based</td>
<td>Automated design (generated from model)</td>
<td>Automated execution</td>
<td>Automated analysis (generated from model)</td>
<td>Automated analysis (generated from model)</td>
</tr>
</tbody>
</table>
Model-Based Testing (MBT)

- **Black-box testing technique**, i.e. functional testing
- **Input**
  - Model of the system under test (SUT)
  - Test selection criteria
- **Output**
  - Derives test cases from the model, based on the test selection criteria
  - Traceability matrix, test coverage, and other test generation information
- **MBT automates:**
  - Test **case design** (generated from model)
  - Test **execution**
  - Test **coverage analysis** (generated from model)
  - Test **result analysis** (generated from model)
Model-Based Testing: An Evaluation

1. Model of SUT
2. Test case generation
3. Generate test code (TCL) and test harness implementation
4. Test execution
5. Test execution analysis (verdict and log)
Thesis Project

- **Feasibility study of model-based testing**
  - Test object: client protocol module of ATM (Automated Teller Machine) client-server system
  - MBT tool: Qtronic
  - Evaluation: Qualitative analysis
  - 4 experiments: based on incremental development

- **Conclusions:**
  - MBT automates generation of test code, test coverage analysis and test result analysis
  - Increases level of abstraction for testing
  - Supports incremental development
Test Object

- Simplified model for an ATM (Automated Teller Machine) client-server system
- Test object limited to the protocol module of the client
  - Defined by a finite state machine
  - Authentication, account balance, withdrawal
Qtronic

- **Tool for automatically designing & generating black-box tests**
  - Stand-alone version
  - Eclipse plugin

- **Input: Model of the system under test**
  - QML: Qtronic Modeling Language used to create models
    - Textual notation: Based on Java
    - Graphical notation: Based on UML
  - Qtronic Modeler
    - Separate modeling tool for creating graphical models

- **Output: Test cases and test code**
  - Generates abstract test cases
  - Generates test code from the abstract test cases
Testing with Qtronic

- Working process
  - Development of test object
  - Model development
  - Test generation and test code (TCL) generation
  - Test harness implementation (glue code) using TCL
  - Test execution

Model-Based Testing: An Evaluation
Experiments

- **Experiment 1: Initial specification**
  - Goal: Apply model-based testing on the test object and execute generated tests against that test object

- **Experiment 2: Extended specification**
  - Goal: Add requirements to see how the implication of an extended model propagates through the process

- **Experiment 3: Modified specification (authentication)**
  - Goal: Change the requirements to see how the implication of model modifications propagates through the process

- **Experiment 4: Logic implemented in test harness**
  - Goal: Evaluate the implications of moving logic from the model to the test harness
## Experiment Results

- **Result table for the 4 experiments**

<table>
<thead>
<tr>
<th>Measures</th>
<th>Exp 1</th>
<th>Exp 2</th>
<th>Exp 3</th>
<th>Exp 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modeling time</strong></td>
<td>2 days</td>
<td>1 day</td>
<td>4 hours</td>
<td>2 hours</td>
</tr>
<tr>
<td><strong>Test generation time</strong></td>
<td>13 s</td>
<td>2 min 34 s</td>
<td>3 min 11 s</td>
<td>3 min 6 s</td>
</tr>
<tr>
<td>Test design configuration coverage</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Number of generated test cases</td>
<td>25</td>
<td>52</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>Time to implement test harness</td>
<td>2 days</td>
<td>1 hour</td>
<td>10 min</td>
<td>2 hours</td>
</tr>
<tr>
<td><strong>LOC: Test suite</strong></td>
<td>2860</td>
<td>6278</td>
<td>7540</td>
<td>7476</td>
</tr>
<tr>
<td>Number of test harness procedures</td>
<td>18</td>
<td>26</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td><strong>LOC: Test harness</strong></td>
<td>99</td>
<td>155</td>
<td>165</td>
<td>229</td>
</tr>
<tr>
<td>Average: LOC / Harness procedure</td>
<td>5.5</td>
<td>5.96</td>
<td>5.89</td>
<td>8.18</td>
</tr>
<tr>
<td>LOC: Test execution environment</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
</tbody>
</table>
Conclusions from the Experiments

- **Modeling**
  - The most time-consuming and challenging task

- **Test generation**
  - Modeling time, test generation time and generated LOC illustrate its gain

- **Test harness**
  - Incrementally developed: manually (empty procedures generated)
  - Procedures for sending input and receiving output from the SUT

- **Test execution environment**
  - Initial effort: required for test harness implementation

- **Test execution resulted in:**
  - Pass/fail verdict for each test case
  - Deadlocks: model and SUT not consistent
  - Output mismatches: expected and actual output differed
Final Comments

- **Conclusions**
  - MBT automates generation of test code, test coverage analysis and test result analysis
  - Increases level of abstraction for testing
  - Supports incremental development
  - Different skills required compared to traditional testing

- **Project a success**: evaluated the MBT concept
  - Limitation: Scalability of the project
- Model quality important
  - Model: the key testing artifact
- **Recommendation**: further evaluation of MBT
Thank You!

- **Any Questions?**

- **Contact:** johan.nordholm24@gmail.com
Qtronic

Model-Based Testing: An Evaluation
Model-Based Testing: An Evaluation

QML Model

- **Wait T-Connect**: netIn:TConnect [msg.type=="Conf+"
 & msg.header=="01"; authError=true; sendErrorMsg("Connection not established")]
Test Design Configuration

- Example of test selection criteria in Qtronic

<table>
<thead>
<tr>
<th>Testing Goals</th>
<th>*Test Design Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Flow</td>
<td>-</td>
</tr>
<tr>
<td>Conditional Branching</td>
<td>-</td>
</tr>
<tr>
<td>State Chart</td>
<td>0</td>
</tr>
<tr>
<td>States</td>
<td>0</td>
</tr>
<tr>
<td>Transitions</td>
<td>-</td>
</tr>
<tr>
<td>2-Transitions</td>
<td>-</td>
</tr>
<tr>
<td>Implicit Consumption</td>
<td>-</td>
</tr>
<tr>
<td>Dynamic Coverage</td>
<td>-</td>
</tr>
</tbody>
</table>
Qtronic Test Generation

- Qtronic console window

Model-Based Testing: An Evaluation
Qtronic Test Case View

- Example of the Qtronic test case view
Traceability Matrix

- Traceability matrix of SIP User Agent Client example

<table>
<thead>
<tr>
<th>Testing Goals</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirements</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.2.2.4 2xx Responses</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UAC core establishes session with ACK</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.1 Terminating a session</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UAC core terminates a session by sending BYE</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UAS core sends OK in response to BYE</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.1.1.2 INVITE timers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resends INVITE after A timeout</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminates INVITE cycle after B timeout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>17.1.2.2 Non-INVITE timers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resends BYE after E timeout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resends CANCEL after E timeout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Terminates BYE cycle after F timeout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Terminates CANCEL cycle after F timeout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Test Case Example

```bash
proc "TC1" {} {
    traceprint "Running test case 'TC1"
    traceprint "Description: This test case covers the following high level requirements:
    traceprint " - requirement: netIn/Authentication/TConnect Conf-
    traceprint "Action: Tester sends inbound event CardInserted to port userIn
    set bankId_1 5500
    set cardNumber_2 100001
    userInCardInserted $bankId_1 $cardNumber_2
    traceprint "Action: SUT is expected to response with outbound event TConnect from port netOut"
    set type_3 "Req"
    netOutTConnect $type_3
    traceprint "Action: Tester sends inbound event TConnect to port netIn"
    set type_4 "Conf-
    netInTConnect $type_4
    traceprint "Covered requirement: requirement: netIn/Authentication/TConnect Conf-
    traceprint "Action: SUT is expected to response with outbound event ErrorMsg from port userOut"
    set msg_5 "Error message: Connection not established"
    userOutErrorMsg $msg_5
}
```

Model-Based Testing: An Evaluation
Test Harness Example

```tcl
proc userOutDepositInfo { money } { \
    global message expected received
    set expected "Deposit info / $money"
    vwait received
}

proc userInAmountInput { money } { \
    global sockChan
    set msg "Amount input / $money"
    send $sockChan $msg
}

proc netInTDisconnect { type } { \
    global sockChan
    set msg "T-Disconnect $type"
    send $sockChan $msg
}
```

Model-Based Testing: An Evaluation
Test Execution

- Execution of test cases

```plaintext
% tclsh test.tcl
Test Case 1: passed
Test Case 2: passed
Test Case 3: passed
Test Case 4: passed
Test Case 5: passed
Test Case 6: passed
Test Case 7: passed
Test Case 8: passed
Test Case 9: passed
Test Case 10: passed
Test Case 11: passed
Test Case 12: failed
Test Case 13: passed
%```
Test Execution Log

Running test case 'TC1'
Description: This test case covers the following high level requirements:
- requirement: netIn/Authentication/TConnect Conf-

Action: Tester sends inbound event CardInserted to port userIn
+SUT input: Card inserted / 5500 / 100001

Action: SUT is expected to response with outbound event TConnect from port netOut
-Expected output: T-Connect Req
-Actual output: T-Connect Req

Action: Tester sends inbound event TConnect to port netIn
+SUT input: T-Connect Conf-
Covered requirement: requirement: netIn/Authentication/TConnect Conf-

Action: SUT is expected to response with outbound event ErrorMsg from port userOut
-Expected output: Error message: Connection not established
-Actual output: Error message: Connection not established

# Test Case passed