Model-based test generation

Zoltan Micskei, Istvan Majzik

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group
Main topics of the course

- Overview (1)
  - V&V techniques, Critical systems

- Static techniques (2)
  - Verifying specifications
  - Verifying source code

- Dynamic techniques: Testing (7)
  - Developer testing, Test design techniques
  - Testing process and levels, Test generation, Automation

- System-level verification (3)
  - Verifying architecture, Dependability analysis
  - Runtime verification
Learning outcomes

- Illustrate how models can be used in testing (K2)
- Explain the typical model-based test generation process (K2)
- Apply different selection criteria to finite state machines to select test cases (K3)
- Use an MBT tool to generate test cases (K3)
What is model-based testing?

“Testing based on or involving models” [ISTQB]

- Not just test generation
- Not just automatic execution
- Not just for model-driven engineering

Source of definition: ISTQB. “Foundation Level Certified Model-Based Tester Syllabus”, Version 2015
Landscape of MBT goals

Shared understanding
  Checking specifications
  Simulation
  Test data creation
  Tests fully executable

more informal -> more formal
Using models in testing (examples)

Behavior of SUT

timer t;
t.start(5.0);
alt {
  [] i.receive("coffee") {
    Count := Count+1;
  }
  [] t.timeout {} }

Test sequences

Test configuration

Source: OMG UTP
Benefits of using models

- **Close communication** with stakeholders
  - Understanding of domain and requirements

- **Early testing**: modeling/simulation/generation

- **Higher abstraction level** (manage complexity)

- **Automation** (different artefacts)
More specific meaning: Test generation

„MBT encompasses the processes and techniques for
- the automatic derivation of abstract test cases from abstract models,
- the generation of concrete tests from abstract tests,
- the manual or automated execution of the resulting concrete test cases”

Typical MBT process

Create test model using FSMs

Use GraphWalker to generate test sequences

Write adaptation to connect to Java code
Example: Model driven workflow

MBT PROCESS
Typical MBT process

Questions for modeling

- **What to model?**
  - What is the test object?
  - Functionality / performance factors / ...

- **What abstraction level to use?**
  - Too many or too few details
  - Separate models for different test objectives

- **What modeling language to use?**
  - Structural, behavioral
Focus of the model

System
- System and intended to be
- Conformance of model-SUT

Usage
- Model environment/users
- Inputs to the system

Test
- Model one or more test case
- E.g. sequences + evaluation
Reuse: Development and Test modeling

What if I have existing design models?

**Problem:** what do we test here?

**Approach:** separate dev. and test models

Typical MBT process

Typical test selection criteria

- **Coverage-based**
  - Requirements linked to the model
  - MBT model elements (state, transition, decision...)
  - Data-related (see spec. test design techniques)

- **Random / stochastic**

- **Scenario- and pattern based** (use case...)

- **Project-driven** (risk, effort, resources...)
Select test cases for full
- requirement coverage
- state coverage
- transition coverage
Typical MBT process

Test generation methods (sample)

- Direct graph algorithms
  - Transition coverage → “New York Street Sweeper problem”

- FSM testing
  - Homing and synchronizing sequences, state identification and verification, conformance...

- LTS testing
  - Equivalence and preorder relations, ioco

- Using model checkers

- Fault-based (mutation)
Typical MBT process

Abstract and concrete test cases

- **Abstract test case**
  - Logical predicate instead of values (e.g. SLOW/FAST instead of 122.35)
  - High-level events and actions

- **Concrete test case**
  - Concrete input data
  - Detailed test procedure (manual or automatic)
Adaptation (automatic execution)

- Adaptation layer
  - Code blocks for each model-level event and action
  - Wrapper around the SUT

- See: Keyword-driven testing
Summary: Taxonomy of MBT approaches

TOOLS AND CASE STUDIES
Typical use cases

- **Fast & easy**
  - Simple modeling
  - Using open tools

- **Full fledged**
  - Complex, commercial tool
  - Full lifecycle support

- **Advanced**
  - Custom modeling languages/tools
MBT tool chain

Source: ISTQB syllabus
Open source tool: GraphWalker

- FSM modell + simple guards
- Coverage: state, transition, time limit (random walk)
- Traversing the graph: random, A*, shortest path
- Generating JUnit test stubs (adapter)

Source: GraphWalker
Industrial MBT tool – Conformiq

- State machine models + Java action code
- Coverage: requirement, state, transition...
- Integration with numerous other tools

Industrial MBT tool – SpecExplorer

- C# model program + adapter code
- Slicing: scenarios, action patterns

Source: https://visualstudiotgallery.msdn.microsoft.com/271d0904-f178-4ce9-956b-d9bfa4902745
Tools (cont’d)

- **CertifyIt (Smartesting)**
  - UML + OCL models

- **MoMuT::UML (academic)**
  - UML state machines, mutation testing

List of tools: [http://mit.bme.hu/~micskeiz/pages/modelbased_testing.html](http://mit.bme.hu/~micskeiz/pages/modelbased_testing.html)
MBT User Survey 2014

~100 participants
32 questions

- “approx. 80h needed to become proficient”
- MBT is effective
- Lots of other details!

Case study: Spotify

Modell + GraphWalker

MBT + test automation

Test automation and Model-Based Testing in agile dev cycle @ Spotify, UCAAT 2013
Case study: MS protocol documentation

- 250+ protocol, 25,000+ pages documentation
- 250+ man year, 350+ engineer
- Tool: SpecExplorer

Details: [http://queue.acm.org/detail.cfm?id=1996412](http://queue.acm.org/detail.cfm?id=1996412)

“Cheat sheet” for introducing MBT

From Robert V. Binder (http://robertvbinder.com/)

<table>
<thead>
<tr>
<th><strong>Recommended</strong></th>
<th><strong>Not recommended</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex SUT behavior</td>
<td>Simple functionality</td>
</tr>
<tr>
<td>Abstractable requirements</td>
<td>Subjective evaluation</td>
</tr>
<tr>
<td>Testable interfaces</td>
<td>Monolithic GUI</td>
</tr>
<tr>
<td>Must to regression testing</td>
<td>Low-value, deprecated GUI</td>
</tr>
<tr>
<td>Sophisticated test engineers</td>
<td>Little or no established testing</td>
</tr>
<tr>
<td></td>
<td>Non-technical QA team</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ISTQB® FOUNDATION LEVEL MODEL-BASED TESTER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction to Model-Based Testing</strong></td>
</tr>
<tr>
<td>Objectives and Motivations for MBT</td>
</tr>
<tr>
<td>MBT Activities and Artifacts</td>
</tr>
<tr>
<td>Integrating MBT into the Software</td>
</tr>
<tr>
<td>Development Lifecycles</td>
</tr>
<tr>
<td><strong>MBT Modeling</strong></td>
</tr>
<tr>
<td>MBT Modeling activities</td>
</tr>
<tr>
<td>Languages for MBT Models</td>
</tr>
<tr>
<td>Good Practices for MBT Modeling Activities</td>
</tr>
<tr>
<td>**Selection Criteria for Test Case</td>
</tr>
<tr>
<td>Generation**</td>
</tr>
<tr>
<td>Classification of MBT Test Selection</td>
</tr>
<tr>
<td>Application Criteria</td>
</tr>
<tr>
<td><strong>MBT Test Implementation and Execution</strong></td>
</tr>
<tr>
<td>Specifics of MBT Test Implementation</td>
</tr>
<tr>
<td>Activities of Test Adaptation in MBT</td>
</tr>
<tr>
<td><strong>Evaluating and Deploying an MBT Approach</strong></td>
</tr>
<tr>
<td>Evaluate an MBT Deployment</td>
</tr>
<tr>
<td>Manage and Monitor the Deployment of an</td>
</tr>
<tr>
<td>MBT Approach</td>
</tr>
</tbody>
</table>

Source: ISTQB
Summary

Many models, test goals and tools

MBT = using models in testing

Scaling from brainstorming to fully automatic test case generation