Reusing Test Cases on Different Levels of Abstraction in a Model Based Development Tool

25.03.2012, Tallinn, Estonia

Jan Olaf Blech            Dongyue Mou            Daniel Ratiu

fortiss GmbH
An-Institut der Technischen Universität München
Motivation

- Manufacturers
  - Provide an abstract specification
  - With abstract test-cases

- Sub-contractors
  - Responsible for development
  - Refine and implement the specification

How to use abstract test-cases in a concrete context?
Modeling Environment

- **AutoFocus**
  - Seamless model-based development tool
  - Using stream processing function concept
  - Binding formal techniques together
    - Modular Component
    - State machines
    - Mode automaton
    - Temporal logics
    - Requirements analysis
    - Test suite generation
    - ....
AutoFocus 3

- Web Site
  - http://af3.fortiss.org
  - Current Version: AutoFocus Phoenix 2.1-RC2 (15.03.2012)
Approach Overview

Abstract Test Case

- Abstract Test Input
  - Abstract Model
  - Abstract Expected-Result

- Representation Function f
  - Concrete Test Input
  - Concrete Model
  - Concrete Expected-Result

- Interpretation Function g
  - Concrete Test Case
Differences from Abstract to Concrete Model

- Reduced behavior space
  - Abstract model contains more execution paths
  - Concrete model may only support a subset of all paths

- Increased complexity
  - Abstract state is normally very simple
    » Boolean value, Abstract token, ...
  - Concrete state is much more complex
    » 32bit integer, list type, ...
Abstract Model

- Focus on limited logical behavior
- No implementation details

Advantages
- Easy to model
- Easy to validate
- Easy to understand
Example – Abstract Model

Text Requirement

- Braking by the driver shall deactivate the ACC function at least if the driver initiated brake force demand is higher than the ACC initiated brake force. (ISO 15622)
1. Formal Requirement

- UserBrake : int
- AccBrake : int

UserBrake > ACCBrake -> State = ACC_Standby

UserBrake <= ACCBrake -> State = *
Concrete Model

- Implementation
  - Refines the specification
  - Under multiple constraints
  - Adds substantial complexity
Example – Concrete Model

- ACC System Model
Example – Concrete Model (2)

- State Automaton for Acceleration Control
Refinement Functions

Transform the abstract inputs to concrete inputs

Representation Function f

Concrete Test Input

Abstract Input

Interpretation Function g

Concrete Output

Transform the abstract outputs to concrete outputs

Expected-Result

Abstract Expected-Result
Example - Refinement Functions

[Diagram showing representation and interpretation functions with variables and states]
Example - Results

- Expected Results
  - Combined view of abstract and concrete test data

<table>
<thead>
<tr>
<th>Abstract</th>
<th>Concrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case 0</td>
<td>21</td>
</tr>
<tr>
<td>Test Case 1</td>
<td>51</td>
</tr>
<tr>
<td>Test Case 2</td>
<td>78</td>
</tr>
<tr>
<td>Test Case 3</td>
<td>100</td>
</tr>
<tr>
<td>Test Case 4</td>
<td>92</td>
</tr>
<tr>
<td>Test Case 5</td>
<td>100</td>
</tr>
<tr>
<td>Test Case 6</td>
<td>33</td>
</tr>
<tr>
<td>Test Case 7</td>
<td>40</td>
</tr>
<tr>
<td>Test Case 8</td>
<td>64</td>
</tr>
<tr>
<td>Test Case 9</td>
<td>34</td>
</tr>
</tbody>
</table>

Test Suite Editor
Conclusion

- Proposed and implemented a method
  - To reuse test cases cross abstract and concrete domains
  - Part of a solution to support development with a manufacturer/sub-contractor relation

Future Work

- Automatic test suite generation for formal requirements
- Improvement of refinement functions for expressing of timing constraints and macro behaviors
Thank You

Dongyue Mou
fortiss GmbH – An-Institut der Technischen Universität München
Guerickestr. 25 | 80805 München | Germany
Tel. +49 89 360 35 22 – 19 | Fax +49 89 360 35 2250
mou@fortiss.org | www.fortiss.org