Automated Requirements-Based Testing

Tuesday, October 7th 2008

Dr. Marc Segelken
Senior Application Engineer
Overview

- Purposes of Testing
- Test Case Generation
- Structural Testing vs. Requirements-based Testing
- Testing Methods
- Requirements Formalization
- Using Formalized Requirements
  - Proving Properties
  - Measuring Requirements Coverage
  - Generating Requirements-based Tests
- Summary
Testing

1. Design Verification
   - Textual Requirements
   - Executable Specification
     - Model used for production code generation
   - Model used for production code generation
     - C Source Code
       - Executable Object Code

2. Code Verification
   - Model Testing
     - Verify that design meets requirements
   - Code Testing
     - Verify that the behavior of source code and object code matches the model
Testing Functional Equivalence between Model and Code

Pairwise equality check
Test Generation for Modules

Overview

- **Input**
  - Model
  - Coverage metric

- **Results**
  - Harness model
  - Input test signals
  - Unreachable objects
  - Detailed reports

- **Optional Input**
  - Modifiable parameter sets
  - Existing coverage data

Decision coverage
Condition coverage
MC/DC
Custom Objectives
Structural Coverage vs. Requirements Coverage

- **DO-178B approach**
  - Find tests for requirements until complete structural coverage (MC/DC) is given

- Structural coverage criteria itself often considered less relevant
  - Requirements coverage unknown
  - since tests for structural criteria usually do not match requirements

- But: No metrics for requirements coverage standardized yet
Map of Testing Methods

Degree of automation

Test Generation covering structure
Metrics
Structural Coverage

Test generation environment constraint
Random Test
(Monte Carlo)

Abstract checks
(limits, invariants)

Test Analysis
(observer based)

Regression Tests
(expected values)

Test Design
(manual)

Manual Review
(each test run)

Formalized Requirements

Relevance for requirements

Manual Review
(each test run)
Formalizing Requirements

- Various dedicated languages (most with certain limitations)
  - Textual: temporal logics (CTL, LTL, …)
  - Graphical: timing diagrams, sequence diagrams, …

- General Language: Same as modeling language
  - e.g., Simulink, Stateflow, MATLAB, …
  - limited to safety requirements (>99% of all industrial requirements)

Simplest Requirement:

Never enter failure state.
Advantages of Formalized Requirements

- Unambiguous precise definition
- Automatic decision if a test run was successful
- Metrics for requirements coverage
- Possibility of automatic test generation
- Possibility to prove correctness
How to use Formalized Properties

Module under Design: 
- In1, In2, In3, In4
- Out1, Out2, Out3, Out4

Terminator: brake, throttle, speed

Property: brake -> throttle decrease
Proving Design Properties

Prove that design meets the key functional requirements
Property Proving

- Inputs
- Results
  - Proof satisfied
  - Proof falsified
    - Counterexamples
    - Model harness
    - Detailed reports

- Optional
  - Assumptions
  - Model parameter ranges/sets
Formalizing Requirements

- Using basic primitives
  - Proof Objective (what is to be proven)
  - Proof Assumption (under what environmental restrictions)

- Or higher level (customer specific) blocks to specify more complex properties
Proving of Properties
Simulink Design Verifier

- **Benefits**
  - Quickly find incorrect behavior
  - Certify correct behavior
  - No stronger approval of correctness possible or necessary
Why using formalized requirements for tests, if proving is much stronger?

- Test cases needed nevertheless for a variety of reasons:
  - Increased confidence in property proving results
  - Helps understanding semantics of formalized requirements
  - Requirements-based tests required

- If proving is not possible, still metrics available
Formalized Properties for Testing

- **Input**
  - Test objectives only in here

- **Results**
  - Harness model
  - Input test signals
  - => Requirements-based tests
Requirements-based Test Generation

- Simple example: Using customer-specified test objectives for desired scenarios

- Tests for path coverage for formalized requirements can be generated automatically
Example

Requirement:
Iteratively, whenever In5 is true, In6 has to occur within next 8 steps

- Path coverage of requirement model == requirement coverage
  - => requirements coverage can be measured
  - => requirements-based tests can be generated automatically
Finding relevant Test Cases no One has thought about …

- Example: Invariant vs. Initial
Generating longer varying test cases

- Using unrolled paths for several iterations
Summary

- Formalized Requirements help to solve various problems
- Possibly serving as test metrics
- Can be used for automatic test case generation