Automated Testing of Embedded Software

Test Automation 2003

Douglas Hoffman
Software Quality Methods, LLC.
24646 Heather Heights Place
Saratoga, California 95070-9710
Phone 408-741-4830
Fax 408-867-4550
doug.hoffman@acm.org
www.SoftwareQualityMethods.com

Copyright © 2003, Software Quality Methods, LLC. No part of these graphic overhead slides may be reproduced, or used in any form by any electronic or mechanical duplication, or stored in a computer system, without written permission of the author.

Topics

• Background
  – Software testing
  – Test automation
  – Test oracles

• Embedded software

• Automation techniques
Running A Software Test

- Test setup
  - SUT program state
  - Data values
  - System environment
- Run test exercise
- Capture/compare actual with expected results

Expanded Testing Model

Test Inputs
Precondition Data
Precondition Program State
Environmental Inputs
System Under Test
Test Results
Postcondition Data
Postcondition Program State
Environmental Results
(Hoffman’s) Fully Automated Software Tests

- Able to run two or more specified test cases
- Able to run a subset of the automated test cases
- No intervention needed after launching tests
- Automatically set-up and/or record relevant test environment
- Run test cases
- Capture relevant results
- Compare actual with expected results
- Report analysis of pass/fail

Testing Analogy: Pulling Weeds

This analogy was first presented by Brian Marick. These slides are from work by James Bach.
Repeating Your Path Won’t Clear the Weeds

Why Automate Tests?

Can help:
- Repeatability
- Streamlining
- Consistency
- Leverage
- Scope

Will not help:
- Faster test development
- Fewer resources
- Strong regression tests
- New tests
- Sufficiency
Testing With An Oracle

Oracle Strategies for Verification

<table>
<thead>
<tr>
<th>No Oracle</th>
<th>True Oracle</th>
<th>Consistency</th>
<th>Self Referential (SVD)</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>- Doesn't check</td>
<td>- Independent</td>
<td>- Embeds answer</td>
<td>- Verifies some</td>
</tr>
<tr>
<td></td>
<td>correctness</td>
<td>generation of</td>
<td>within data in the</td>
<td>characteristics of</td>
</tr>
<tr>
<td></td>
<td>of results</td>
<td>all expected</td>
<td>messages</td>
<td>values</td>
</tr>
<tr>
<td>Advantages</td>
<td>- Can run any amount</td>
<td>- All encountered</td>
<td>- Fastest method using</td>
<td>- Faster and easier</td>
</tr>
<tr>
<td></td>
<td>of data (limited only</td>
<td>errors are detected</td>
<td>an oracle</td>
<td>than True Oracle</td>
</tr>
<tr>
<td></td>
<td>by the time the SUT</td>
<td></td>
<td>- Verification is</td>
<td>- Often much less</td>
</tr>
<tr>
<td></td>
<td>takes)</td>
<td></td>
<td>straightforward</td>
<td>expensive to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Can generate and</td>
<td>create and use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>verify large amounts</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of data</td>
<td></td>
</tr>
<tr>
<td>Disadvantages</td>
<td>- Only spectacular</td>
<td>- Expensive to</td>
<td>- Original run may</td>
<td>- Can miss</td>
</tr>
<tr>
<td></td>
<td>failures are noticed</td>
<td>implement</td>
<td>include undetected</td>
<td>errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Complex and often</td>
<td>errors</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>time-consuming when run</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Douglas Hoffman  Copyright © 2003, SQM, LLC.
What is Embedded Software?

- Software that is part of a larger system and performs some of the requirements of that system (IEEE 610.12-1990)
- Any software system that must be designed on a platform different from the platform on which the system is intended to be deployed (Encontre, Spring 2002, STA)

Environments

- Simulator
  - Software based
  - Debugger capabilities
- Emulator
  - Hardware & software
- Target Environment
  - Cross compilers
  - Weak debuggers
Test Hooks and Scaffolding

• Purpose:
  – Monitor
  – Control
  – Stimulate

• Environment specific mechanisms
  – Interface with SUT
  – Fill in for co- and sub-processes
  – Rationalizing actions

Oracle Strategies for Embedded Software

• No oracle approach is a poor choice
• True oracle may be available
• Different environments are potential consistency oracles
• Self referential approach usually not applicable
• Heuristic approach may be required
Automation Techniques for Embedded Software

- Scripting
- Frameworks
- Individual test programs

Test Scripting

- User language
- Test development independent from tools
- Separates test exercises from mechanisms
- Tailorable to the context
- Table driven tests are simplest examples
- *Not* capture/playback
Scripting for Embedded Systems

**Pros:**
- Natural language
- Can operate across environments
- Tailorable to contexts
- Can evolve as needed
- Separates test specifications from mechanisms

**Cons:**
- Resource sink
- Environment dependencies
- Limits what is possible or interesting
- Can limit to least common denominator

Frameworks

- Subroutine library
- Common actions
- Complex, repeated actions
- Test development requires programming
**Framework for Embedded Systems**

**Pros:**
- Standardization
- Efficiency
- Enforces discipline
- Can evolve as needed

**Cons:**
- Limits what is possible or interesting
- May be one person’s ideal
- Can limit to least common denominator
- Resource sink
- Maintenance costs

**Individual Test Programs**

- Tests are explicit programs
- Seeking widely different errors
- Independent tests
- Separable test activities
Individual Test Programs

**Pros:**
- Can craft each test
- Test specifications is the mechanism

**Cons:**
- Each test is crafted
- High cost of development
- Strong environment dependencies
- High maintenance costs

Choosing A Strategy

- Understand the development and platform environments
- Evaluate the automation options
- Identify the oracle strategy
- Consider costs and benefits between manual and automated testing
Closing Thoughts

• Testing embedded software can be challenging
• Test automation is hard work
• The combination can be interesting:
  – platforms / environments
  – test creation and maintenance
  – getting useful information
  – oracles and validation

References