Model-Based Statistical Testing

Tom Swain
tomswain@comcast.net
A Statistical Approach
Frames Testing in Science

- The question is not whether to test, but:
  - what to test
  - how much to test
- Statistical science supports rational, economical, and defensible answers to these questions.
Testing is Always Sampling

Population (All Uses)

What to test: a statistically appropriate sample

Sample (Tests)

How much to test: a risk/benefit tradeoff
Statistical Usage Testing

Statistical Usage Testing involves:

- Modeling the population of uses
- Selecting a sample for testing
- Estimating field performance from test sample results
- Making a release decision
Statistical Testing can be used across the Life Cycle

- Development testing assesses evolving product quality
- Certification testing estimates fielded product performance
The Statistical Testing Process
Software Testing Process

- **Usage Modeling**
  - Model Analysis and Validation
  - Test Planning
  - Testing
  - Product and Process Measurement
  - **Software Certification**
Usage Modeling

A usage model is a statistical representation of all possible uses of the system.

- model structure is a directed graph
  - nodes are states-of-use
  - arcs are possible transitions

- usage profile
  - probability of making each transition

- one structure, many profiles
Usage Model: All Possible Uses and their Likelihood

STATE MACHINE
Nodes are states-of-use
Arcs are possible stimuli
Probabilities define expected usage
Test case is path from initial to terminal state
Example:
Two Usage Profiles

One structure

Two sets of transition probabilities

<table>
<thead>
<tr>
<th>From-State</th>
<th>To-State</th>
<th>Uniform Probability</th>
<th>Specific Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Ready</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ready</td>
<td>Entry Error</td>
<td>.25</td>
<td>.2398</td>
</tr>
<tr>
<td>Ready</td>
<td>1_OK</td>
<td>.25</td>
<td>.7193</td>
</tr>
<tr>
<td>Ready</td>
<td>Ready</td>
<td>.25</td>
<td>.0400</td>
</tr>
<tr>
<td>Ready</td>
<td>Alarm</td>
<td>.25</td>
<td>.0010</td>
</tr>
<tr>
<td>Entry Error</td>
<td>Ready</td>
<td>.33</td>
<td>.9590</td>
</tr>
<tr>
<td>Entry Error</td>
<td>Entry Error</td>
<td>.33</td>
<td>.0400</td>
</tr>
<tr>
<td>Entry Error</td>
<td>Alarm and Entry Error</td>
<td>.33</td>
<td>.000001</td>
</tr>
<tr>
<td>1_OK</td>
<td>Entry Error</td>
<td>.2</td>
<td>.2304</td>
</tr>
<tr>
<td>1_OK</td>
<td>Ready</td>
<td>.2</td>
<td>.0096</td>
</tr>
<tr>
<td>1_OK</td>
<td>2_OK</td>
<td>.2</td>
<td>.7200</td>
</tr>
<tr>
<td>1_OK</td>
<td>1_OK</td>
<td>.2</td>
<td>.0400</td>
</tr>
<tr>
<td>1_OK</td>
<td>Alarm and Entry Error</td>
<td>.2</td>
<td>.000001</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2013, Tom Swain and Software Quality Research Laboratory
Software Testing Process

- Usage Modeling
- Model Analysis and Validation
- Test Planning
- Testing
- Product and Process Measurement
- Software Certification

© 2013, Tom Swain and Software Quality Research Laboratory
A Usage Model is a Finite-State Markov Chain

- Well-understood formalism
- Rich body of analytical results
- Engineering basis for testing
- Objectivity in test planning and management
- Simulates “use” of product and not the product itself
Standard Analysis

All standard statistics for Markov chains have meaningful interpretations in the application to software testing.

- Long run occurrence of each state
- Expected number of transitions in a test case
- Expected number of transitions until a particular state first appears
- Estimate of the necessary and sufficient number of test cases to run
Model Revision and Validation

- Analytical results are inescapable, given the model.
- If results do not reflect what is known of the real-world usage, the model must be revised.
- Analyze and revise the model until the it is an acceptable description of expected use.
Generating Models From Systems of Constraints

- An alternative to the analyze-revise cycle.
- Describe the model as a system of constraints.
- Set objective functions to optimize.
- Generate the Markov chain transition matrix as the solution to the system of constraints.
Model Constraint Examples

- One transition is a function of another transition
  \[\text{e.g.: } \text{prob (GoodDigit)} = 3 \text{ (prob (BadDigit))}\]
- A probability is within a range
  \[\text{e.g.: } .1 < \text{prob (BadDigit)} < .3\]
Software Testing Process

- Usage Modeling
- Model Analysis and Validation
- **Test Planning**
- Testing
- Product and Process Measurement
- **Software Certification**
Sampling Options

Population (All Tests)

- Random
- Weighted

Expected Usage

- Non-random
- Structural

Model Coverage

- Contractual Requirements, Industry Standards

© 2013, Tom Swain and Software Quality Research Laboratory
Nonrandom Testing

- Coverage tests
  (cover all arcs at the least cost of testing)

- Mandatory tests
  (policy, contract, regulatory, safety, security)

- Regression tests

- Importance tests
  (generate tests in order of probability or cost)
Random sample testing

- Random test cases are generated from the usage model.
- Random testing permits statistical analysis of the sample and generalization to the population.
- Each test case is a sequence of stimuli, and random test cases may be reused.
Testing Scripts

Script commands are attached to arcs and give the instructions for testing the transition:

- Manual testing
  - written instructions
  - data to use
  - items to check

- Automated testing
  - commands for testing equipment
  - commands for testing software
  - statements in a programming language
Software Testing Process

1. Usage Modeling
2. Model Analysis and Validation
3. Test Planning
4. Testing
5. Product and Process Measurement
6. Software Certification

© 2013, Tom Swain and Software Quality Research Laboratory
Testing

Record all failures by test case number and transition number where failure occurs, then:

- Estimate reliability based on testing experience
- Evaluate stopping criteria
Software Testing Process

- Usage Modeling
- Model Analysis and Validation
- Test Planning
- Testing
- Product and Process Measurement
- Software Certification
Process Measurement

- Statistical testing supports incremental development with feedback on the development process.
- Reliability growth modeling can be applied to incremental development.
- Reliability estimates can be used to project product support effort.
Reliability Measurement

- Test case pass/fail statistics give reliability and confidence based on binomial distribution.

- Bayesian models
  - provide reliability estimates regardless of whether failures are observed
  - allow use of prior reliability information
Software Testing Process

Usage Modeling

Model Analysis and Validation

Test Planning

Testing

Product and Process Measurement

Software Certification
Software Certification

- Certification establishes product conformance with well-defined standards.
- Product certification requires a process that is independently repeatable within statistical variation.
- Statistical testing supports quantitative certification through statistical characterization of system use and reliability.
EXAMPLE:
The Security Alarm
Security Alarm Stimuli

SET activates device

A 3-digit code deactivates device, so possible inputs: GoodDigit, BadDigit

CLEAR erases all digits that have been entered

Trip activates alarm
Security Alarm Responses

SET button
- **Light on** when device is activated
- **Light off** when correct code entered

Alarm siren
- **Alarm on** when tripped
- **Alarm off** when correct code entered
The structure of the usage model (the graph) represents all possible uses of the software.

Transition probabilities on the arcs represents the expected use.
Transition Probabilities on Exit Arcs sum to 1

Probabilities come from historical data or engineering judgment

- **Idle** transitions to **Ready** with probability 1.
- **Ready** transitions to **Entry** with probability .240, to **Alarm** with probability .040, and to **1_OK** with probability .719.
- **Alarm** transitions to **1_OK** with probability 1.000.
- **Entry** transitions to **Alarm** with probability .001.

© 2013, Tom Swain and Software Quality Research Laboratory
Security Alarm Usage Model

- **Idle**
- **Ready**
- **Entry Error**
- **1_OK**
- **2_OK**
- **Alarm & Entry Error**
- **Alarm & 1_OK**
- **Alarm & 2_OK**
- **Term**

**Transitions and Probabilities:**
- S → Idle: 1
- T → Ready: 0.001
- B → Entry Error: 0.24
- G → Entry Error: 0.72
- C → Alarm: 0.96
- T → Alarm: 0.00001
- B → Alarm & Entry Error: 0.23
- G → Alarm & Entry Error: 0.0001
- C → Alarm & 1_OK: 0.88
- B → Alarm & 2_OK: 0.35
- G → Alarm & 2_OK: 0.53
- C → 1_OK: 0.18
- B → 2_OK: 0.18
- G → 2_OK: 0.33
- C → 2_OK: 0.02
- B → Term: 0.23
- G → Term: 0.72
- C → Term: 0.01

Self-loop for other inputs

© 2013, Tom Swain and Software Quality Research Laboratory
Operational Profile from Markov Analysis

<table>
<thead>
<tr>
<th>State</th>
<th>Long run occupancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>0.109597</td>
</tr>
<tr>
<td>Ready</td>
<td>0.270206</td>
</tr>
<tr>
<td>Entry Error</td>
<td>0.152519</td>
</tr>
<tr>
<td>1_OK</td>
<td>0.202452</td>
</tr>
<tr>
<td>2_OK</td>
<td>0.151837</td>
</tr>
<tr>
<td>Alarm</td>
<td>0.001447</td>
</tr>
<tr>
<td>Alarm &amp; 1_OK</td>
<td>0.000868</td>
</tr>
<tr>
<td>Alarm &amp; 2_OK</td>
<td>0.000521</td>
</tr>
<tr>
<td>Alarm &amp; Entry Error</td>
<td>0.000956</td>
</tr>
<tr>
<td>Term</td>
<td>0.109597</td>
</tr>
</tbody>
</table>

These results are validated against data and expert opinion.
### Operational Profile from Markov Analysis

<table>
<thead>
<tr>
<th>From State</th>
<th>Arc</th>
<th>Long run occupancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>Set</td>
<td>0.12308685</td>
</tr>
<tr>
<td></td>
<td>BadDigit</td>
<td>0.07275884</td>
</tr>
<tr>
<td></td>
<td>GoodDigit</td>
<td>0.21827653</td>
</tr>
<tr>
<td></td>
<td>Set, Clear</td>
<td>0.01212647</td>
</tr>
<tr>
<td></td>
<td>Trip</td>
<td>0.00030316</td>
</tr>
<tr>
<td>Ready</td>
<td>Clear</td>
<td>0.16443191</td>
</tr>
<tr>
<td></td>
<td>Set, BadDigit, GoodDigit</td>
<td>0.00685811</td>
</tr>
<tr>
<td></td>
<td>Trip</td>
<td>0.00000171</td>
</tr>
<tr>
<td>Entry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These results are validated against data and expert opinion.
Operational Profile from Markov Analysis

<table>
<thead>
<tr>
<th>Stimulus</th>
<th>Long run occupancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>0.13900261</td>
</tr>
<tr>
<td>GoodDigit</td>
<td>0.50644176</td>
</tr>
<tr>
<td>BadDigit</td>
<td>0.16537277</td>
</tr>
<tr>
<td>Clear</td>
<td>0.16937841</td>
</tr>
<tr>
<td>Trip</td>
<td>0.00030885</td>
</tr>
<tr>
<td>All self-loops</td>
<td>0.0195</td>
</tr>
</tbody>
</table>

These results are validated against data and expert opinion.
Statistical Tests

- Random tests
  random scenarios based on transition probabilities

- Importance tests
  scenarios that traverse “most important” arc sequences, where importance may be related to cost, time, probability, etc.
Sample Security Alarm Test Script

# ==============================================================
# Trajectory: 0  
# Model: Security Alarm  
# Key:  
#  
# Events: 14  
# ==============================================================
# Step: 1, Trajectory: 0  
[S0]."S"  
# Step: 2, Trajectory: 0  
[S1]."G"  
# Step: 3, Trajectory: 0  
[S3]."B"  
...  
# Step: 13, Trajectory: 0  
[S6]."G"  
# Step: 14, Trajectory: 0  
[S7]."G"
## Test Case Interleaving

<table>
<thead>
<tr>
<th>Test Case A</th>
<th>Test Case AB</th>
<th>Test Case B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>B1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>B2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>A2</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>A3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>B3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A5</td>
<td></td>
</tr>
</tbody>
</table>
Interleaving Uses

- Servers to Multiple Clients
- Multiplexed Input Streams
- Non-modal GUI
Test Automation

- Many automated test tools already exist
- Convert test cases to test scripts which can be executed by a test tool
- Associate test instructions with arcs and states in a model
Test Automation

- Parametric Data Generation for Abstract Stimuli
- Partition and Sample Input Space
- Test Execution
- Compare Results to Oracle
- Record Pass/Fail
- Result Checking and Comparison
- Analyze Results
- Test Case Generation
- Generate Test Cases
- Export Test Scripts
- Test Record
- Usage Model
- Release?
Test Oracle

- Must be able to decide pass / fail for each test.
- Automation techniques:
  - Build oracle into test script
  - Use self-checking data
  - Use known cases
  - For some cases, let a human decide
Test Record Management

- Record pass/fail information for each test case
  - Specify arc(s) where failure occurred
  - Specify if and where test case aborted
Security Alarm Test Record

**SecurityAlarm_1.str,5,19**
On the test case 1 for model SecurityAlarm, failures occurred on events 5 and 19, but testing continued to the end of the test.

**SecurityAlarm_2.str,17,21,S**
On the test case 2 for model SecurityAlarm, failures occurred on events 17 and 21, and testing stopped.
Test Results Analysis

- State/Stimulus/Arc coverage
- Stimulus/Arc reliability
- Single use reliability
Benefits of Model Based Testing

- Better Product
  - Clearer requirements, improved specification
- Better Use of Resources
  - Optimization of testing strategy
  - Reusable assets: models, test plans, scripts, test cases
- Shorter Life Cycle
  - Test planning done in parallel with development
  - Easier test automation
- Better Management
  - Quantitative support for management decisions
  - Quantification of expected field reliability
  - Clear path to continuous process improvement