Risk Measurement for the Real (and Imperfect) World

Darryl Nicholson
ContactDarrylNicholson@gmail.com
Agenda

- Introduction
- Context / Background
- The Problem
- Scenarios & Calibration
- Scenario Lifecycle
- Deliverable
- Questions
Introduction

- Who am I and why am I here?

- Risk and Regression Testing
  - Calibration of test plans
  - Minimalistic approach to deliver sw quickly
  - Our methods for Risk management
  - Designed to drive revenue
  - Fights natural instincts to be policeman/gatekeepers

- MRS – Minimum Regression Set
  - Our implementation of Code coverage
  - Controversial
SaaS Environment
- Our clients dictate schedules to sell services we build
- Hybrid SOA Production environment
- Process Billions of $$ in payments
- We are built for speed; wired for changes.

Speed to Market is key
- Caution doesn’t pay the bills
- Compensation comes from driving revenue
- Cost to fix a Production bug is roughly equal to QA
Continuous Test Case Growth

- Customer review cycles and feedback
- New Clients & New Features
- Innovation in our product portfolio
- SOA enhancements that magnify the test problem
- Production test escapes
The Problem

- Result: Continuous test case growth in an unstructured quasi-subjective manner.

- Regression testing burden grows.
  - Each new release cycle needs additional time and/or resources to complete
  - Project Managers, Business Executives, Marketing and Customers never like this answer

- Not sustainable nor scalable
The Solution

- We chose to instrument our test cases using Code coverage techniques.

- Resulting test case set from this analysis is the “Minimum Regression Set” (MRS).

- MRS easily maps to requirements, use cases, feature definitions, etc. All artifacts easily understood by key stakeholders.
Engineering team drives API & Code coverage unit tests with Cobertura

- Engineering has an extensive set of Unit tests that drive MT API’s but do not include the UI
- All feature complete QA releases have an instrumented MT.

- UI: User Interface Layer
- MT: Middle Tier (Java)
- DB: Database
Test Scenarios

- Our clients tend to describe changes in terms of business use cases, marketing ideas or product delivery strategies rather than traditional software requirements.

- Client definition, in whatever form it arrives, is used to describe “Test Scenarios”

- Segregate out the test case data and refer to these elements as “Attributes”.
Test Scenarios

- Process looks like this:

- Example: process credit card transactions from all states for different amounts and payment methods
A typical review for one of our web products will create 700–900 Scenarios.

- Creates Joint Ownership

- Are all defined Scenarios truly needed?
Test Calibration

Test Calibration is the process by which we create an MRS from the large set of Scenarios.

Classify in 3 categories:

- **Cat 1: The MRS.** Single Scenario that exercises a unique code path, is repeatable and measured.

- **Cat 2:** A scenario that does not add code path uniqueness but adds unique data sets based on attributes.

- **Cat 3:** A scenario that has neither code path uniqueness nor adds unique attribute data.
Test Calibration

- MRS Definition of Category 1
- Instrumented MT-JAR file in the System Under Test
- Run each scenario to increase code coverage
Example from Cobertura home page

- Simply run Scenarios and verify coverage is increasing
- Goals: 100% API & code coverage.

### Coverage Report - All Packages

<table>
<thead>
<tr>
<th>Package</th>
<th># Classes</th>
<th>Line Coverage</th>
<th>Branch Coverage</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Packages</td>
<td>55</td>
<td>75%</td>
<td>64%</td>
<td>2.319</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.ant</td>
<td>11</td>
<td>52%</td>
<td>45%</td>
<td>1.848</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.check</td>
<td>3</td>
<td>0%</td>
<td>0%</td>
<td>2.429</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.coveragedata</td>
<td>13</td>
<td>N/A</td>
<td>N/A</td>
<td>2.277</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.instrument</td>
<td>10</td>
<td>90%</td>
<td>75%</td>
<td>1.854</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.merge</td>
<td>1</td>
<td>86%</td>
<td>88%</td>
<td>5.5</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.reporting</td>
<td>3</td>
<td>67%</td>
<td>80%</td>
<td>2.882</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.reporting.html</td>
<td>4</td>
<td>91%</td>
<td>77%</td>
<td>4.444</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.reporting.html,files</td>
<td>1</td>
<td>87%</td>
<td>62%</td>
<td>4.5</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.reporting.xml</td>
<td>1</td>
<td>100%</td>
<td>95%</td>
<td>1.524</td>
</tr>
<tr>
<td>net.sourceforge.cobertura.util</td>
<td>9</td>
<td>60%</td>
<td>69%</td>
<td>2.892</td>
</tr>
<tr>
<td>someotherpackage</td>
<td>1</td>
<td>83%</td>
<td>N/A</td>
<td>1.2</td>
</tr>
</tbody>
</table>
Generally after execution of approximately a third of the defined Scenarios, the code coverage needle will stop incrementing far short of 100% coverage.

This is the moment where we realize that the Scenarios analysis done as an intellectual exercise has missed a number of valid cases.

Validation of the method!
MRS Findings

- Typically what is missed and overlooked:
  - the error handling routines
  - obscure use cases
  - available functionality that was not obvious at review or “Snuck in”

- When running with code coverage enabled, these potential test escapes are very obvious.
After MRS is defined, a final UI Code review is required.
The White space is the UI code structures not measured since their scope is entirely in the UI Framework.

Examples: JQuery elements, Analytic web tags, form validation logic.

These are manually added to the MRS.
Feedback loop
Catch "feature Creep"
Iterative and keeps conversation flowing
Test Escapes

- They happen. Root cause expressed as an MRS

- In our system, test escapes are generally:
  - Automated test failure
  - MRS Definition inaccuracy (missed)
  - White Space analysis incorrect
  - Scenario not executed

- First 3 = MRS additions
- 4th Case is the price of too much speed & Risk
The Deliverable

- We live in an imperfect world.

- Accept – Deliver code with the “Sun & Moon alignment method”

- If we “Have to …” when QA has not finished testing then QA has a simple message for the team  MRS = 45%.
Questions?