Test Suite Prioritization and Reduction by Combinational-based Criteria

Dr. Renée Bryce
Associate Professor
University of North Texas
Renee.Bryce@unt.edu

Presentation outline

- Test Suite Prioritization
  - Exercise: Prioritize a test suite
- Test Suite Reduction
  - Exercise: Reduce a test suite using HGS
- Discussion
Test Suite Prioritization

- **Test Suite Prioritization**
  - Problem: Given T, a test suite, Π, the set of all test suites obtained by permuting the tests of T, and f, a function from Π to the set of real numbers, the problem is to find π ∈ Π such that ∀ π′ ∈ Π, f(π) ≥ f(π′). In this definition, Π refers to the possible prioritizations of T and f is a function applied to evaluate the orderings.

Case Study: Prioritizing User-session-based Test Suites

- **Methodology:** Convert web logs to user-session-based test suites, prioritize, and write to an XML format.

- **Algorithm:** Efficiently prioritize by combinatorial-based coverage for large test suites

- **Empirical Studies:** Families of empirical studies to analyze the effectiveness in relation to characteristics of the applications and test suites.
Research Questions

- Can we improve the rate of fault detection for user-session-based testing with new prioritization criteria?
- Which techniques are valuable in different scenarios?
  - i.e.: tests have a high/low Fault Detection Density
  - i.e.: predicted distribution of faults (deemed from prior versions of the software)
- Can we fine tune the criteria?
  - i.e.: cost-based prioritization

Prioritization Metrics

- Test length based on number of base requests:
  - order by the number of HTTP requests in a test case
- Frequency-based prioritization:
  - order such that test cases that cover most frequently accessed pages/sequence of pages are selected for execution before test cases that exercise the less frequently accessed pages/sequences of pages.
- Unique coverage of parameter-values:
  - order tests to cover all unique parameter-values as soon as possible
- 2-way parameter-value interaction coverage:
  - order tests to cover all pair-wise combinations of parameter-values between pages as soon as possible
- Test length based on number of parameter-value:
  - order by number of parameter-values used in a test case
- Random:
  - randomly permute the order of tests
Empirical Studies

- TerpCalc, TerpPaint, Terp Spreadsheet, and TerpWord
- Online Bookstore
- Online Course Project Manager (CPM)
- Online Conference Management System
- SchoolMate
- Online Music Store
- Metavist (sponsored by USDA)

Results for an on-line system for a Course Project Manager and 890 Test Cases

### Sample results

<table>
<thead>
<tr>
<th>% of test suite run</th>
<th>Most frequent requests</th>
<th>No. of Requests Long to short</th>
<th>No. of Requests Short to long</th>
<th>PVs Long to short</th>
<th>PVs Short to Long</th>
<th>1-way</th>
<th>2-way</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>85.28</td>
<td>78.17</td>
<td>75.14</td>
<td>83.53</td>
<td>16.38</td>
<td>83.79</td>
<td>83.72</td>
<td>48.63</td>
</tr>
<tr>
<td>20</td>
<td>88.52</td>
<td>80.34</td>
<td>77.76</td>
<td>88.77</td>
<td>25.6</td>
<td>87.78</td>
<td>90.8</td>
<td>57.55</td>
</tr>
<tr>
<td>30</td>
<td>89.4</td>
<td>81.77</td>
<td>80.27</td>
<td>88.77</td>
<td>26.44</td>
<td>91.54</td>
<td>91.72</td>
<td>64.51</td>
</tr>
<tr>
<td>40</td>
<td>89.86</td>
<td>84.58</td>
<td>81.39</td>
<td>92.71</td>
<td>28.76</td>
<td>94.79</td>
<td>95.64</td>
<td>69.19</td>
</tr>
<tr>
<td>50</td>
<td>91.04</td>
<td>85.58</td>
<td>82.95</td>
<td>92.71</td>
<td>30.33</td>
<td>94.79</td>
<td>95.64</td>
<td>73.03</td>
</tr>
<tr>
<td>60</td>
<td>91.58</td>
<td>87.14</td>
<td>84.44</td>
<td>94.26</td>
<td>34.64</td>
<td>94.79</td>
<td>95.64</td>
<td>75.37</td>
</tr>
<tr>
<td>70</td>
<td>92.1</td>
<td>87.74</td>
<td>85.15</td>
<td>94.26</td>
<td>39.15</td>
<td>94.79</td>
<td>95.64</td>
<td>77.37</td>
</tr>
<tr>
<td>80</td>
<td>92.35</td>
<td>88.27</td>
<td>86.21</td>
<td>94.26</td>
<td>39.58</td>
<td>94.79</td>
<td>95.64</td>
<td>78.24</td>
</tr>
<tr>
<td>90</td>
<td>92.37</td>
<td>88.3</td>
<td>86.31</td>
<td>94.26</td>
<td>42.18</td>
<td>94.99</td>
<td>95.64</td>
<td>78.45</td>
</tr>
<tr>
<td>100</td>
<td>92.45</td>
<td>88.36</td>
<td>86.35</td>
<td>94.26</td>
<td>43.09</td>
<td>94.99</td>
<td>95.64</td>
<td>78.49</td>
</tr>
</tbody>
</table>

### Test prioritization by interaction coverage

- Test suite prioritization
  - GUI-based testing
Empirical Studies

- Traffic Collision Avoidance System
- GUI-based Testing
  - Word processor
  - Spreadsheet
  - Paint
  - Calculator
- Web application Testing
  - Bookstore
  - Course Project Manager
  - Conference Management Software

Transfer of Work

Potential users that have contacted NIST to use our tool:
- AT&T
- BBC (for Winter Olympics website)
- Booz Allen Hamilton
- Angel.com
- U.S. Army Test and Evaluation Research Laboratory, Aberdeen Proving Ground
- A2Z Research and Development
- NASA IV&V
Transfer of Work (Demo)


Next steps

- Methodologies
  - Examining issues with RIAs

- Algorithms
  - Hybrid techniques

- Empirical Studies
  - “Real” studies
  - RIA studies
Test Suite Reduction

Problem: Given T, a test suite with test cases \{t_1, t_2, t_3, t_4\}, a set of testing requirements, \{r_1, r_2, r_3, r_4\}, that must be satisfied to provide the desired test coverage of the program, and subsets \{T_1, T_2, ..., T_n\} of T, one associated with each of the others such that any one of the tests belonging to \(T_i\) satisfies . Find the *minimal cardinality* subset of T that exercises all of the requirements exercised by the original test suite T.

Original Test Suite
(Too large for our budget)

Reduced Test Suite
(Fits into budget)

Reduction Example

Original Test Suite

- \{t_1, t_2, t_3, t_4\}

Requirements covered by the test suite

- \{r_1, r_2, r_3, r_4\}

Problem: Reduce the test suite such that it maintains coverage of these requirements
Test Suite Reduction Example

<table>
<thead>
<tr>
<th>T</th>
<th>Requirement</th>
<th>Ti</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>{t3, t4}</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>{t4}</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>{t1, t2, t3, t5}</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>{t1, t2, t3}</td>
</tr>
</tbody>
</table>

In this example, there are three possible solutions. We highlighted 1: {t1, t4}

Test Suite Reduction Example

<table>
<thead>
<tr>
<th>T</th>
<th>Requirement</th>
<th>Ti</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>{t1, t5}</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>{t5}</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>{t1, t2, t3}</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>{t3, t6}</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>{t1, t4}</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>{t1, t5}</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>{t3, t5, t7}</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>{t2, t3, t5, t7}</td>
</tr>
</tbody>
</table>

HGS Algorithm

3. T4 is of cardinality 2, there is a tie between t3 and t6, so we look at sets of size cardinality (m+1). We choose t3.

Reduced Test Suite: {t5, t1, t3}
Exercise

Reduce this test suite using the HGS algorithm:

<table>
<thead>
<tr>
<th>T</th>
<th>Requirement</th>
<th>T_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>{t_1, t_5}</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>{t_5}</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>{t_1, t_2, t_3}</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>{t_3, t_6}</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>{t_1, t_4}</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>{t_1, t_6}</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>{t_3, t_4, t_7}</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>{t_2, t_3, t_4, t_7}</td>
</tr>
</tbody>
</table>

Test Suite Reduction Example

HGS Algorithm

3. T_4 is of cardinality 2, there is a tie between t_3 and t_6, so we look at sets of size cardinality (m+1). We choose t_3.

Reduced Test Suite: \{t_5, t_1, t_3\}