Leveraging User-Session Data to Support Web Application Testing

Authors: Sebastian Elbaum, Gregg Rotheermal, Srikanth Karre, and Marc Fisher II

Presented By: Rajiv Jain
Outline

• Introduction
• Related Work
• Testing Techniques
• Experiment and Results
• Additional Considerations
• Conclusion
What is a Web Application?
Introduction

• Web applications
  – Can have large numbers of users
  – Change rapidly
  – Multi-tiered architectures

• Similar to GUIs
  – Event and user driven
  – Conventional testing techniques may not work
Related Work

• Web Testing Techniques
  – Validation / Static analysis
  – Capture + replay tools (Selenium)
  – Conventional white-box testing techniques
    • Unit testing + integration testing
    • Dataflow
    • State-based
    • No work on fault detection, only coverage
White Box Testing Techniques

• Ricca and Tonella
  – Models web application as graph
  – Nodes = Web objects
  – Edges = Relationships between objects
  – Test requirements and cases extracted from graph
  – Requires test engineer to create test cases
White Box Testing Techniques

- Ricca and Tonella
White Box Testing Techniques

• Ricca and Tonella-1 (WB-1)
  – Tests all edges
  – Uses linearly independent paths
  – Ignores circular links

• Ricca and Tonella-2 (WB-2)
  – Boundary values used as input
  – Each condition / All condition
User-Session Testing Techniques

- User session is a TCP session
- Session is made up of several requests
- Request = URL + Name-value pairs
- Transparently collect several user sessions
- Uses sessions used to create test cases
User-Session Testing Techniques

• Direct Reuse of User Sessions (US-1)
  – Analogous to a capture replay tool
• Combining Different User-Sessions (US-2)
  – Randomly combines two user-sessions with overlapping requests
• User Sessions with Form modifications (US-3)
  – Test cases randomly delete one value character
    • One case per name-value pair
    • One case modifying all name-value pairs
Hybrid Testing Techniques

• Combines user-session and white box tests

• Hybrid 1 (HYB-1)
  – Match user-sessions to requirements in WB-1/2
  – If unable to match sessions to requirement, requirement is ignored

• Hybrid 2 (HYB-2)
  – Expands hybrid-1 by creating cases for unmatched requirements
Experiment Setup

• Research Questions
  – How effective are the techniques
  – Does technique appropriateness vary with fault type?
  – Relationship between number of user sessions and test suite effectiveness

• Independent Variable = 7 test techniques
• Dependent Variable = Coverage, Fault Detection
Experiment Setup

• Test subject is an online bookstore
  – Implemented in Perl (67 functions, 399 blocks)
  – Uses MySQL (7 tables)
  – Hosted on a Apache web server

• Fault seeding was used
  – 50 “realistic” faults added by 2 grad students
    • Scripting faults
    • Form faults
    • Database query faults
Experiment Setup

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Suite Size</td>
<td>28</td>
<td>64</td>
<td>85</td>
<td>84</td>
<td>407</td>
<td>1004</td>
<td>1089</td>
</tr>
<tr>
<td>Requests</td>
<td>99</td>
<td>241</td>
<td>1975</td>
<td>1919</td>
<td>2742</td>
<td>1428</td>
<td>1397</td>
</tr>
</tbody>
</table>

- Test Suite Creation
  - White box
    - 75 hours spent creating the representation model
    - Completed prior to fault seeding
  - User-session
    - 73 users navigated the website using IE
    - Sessions were recorded with Apache/Javascript
  - Oracle
    - Web application output prior to fault seeding
## Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>abs</td>
<td>%</td>
<td>abs</td>
<td>%</td>
<td>abs</td>
<td>%</td>
<td>abs</td>
</tr>
<tr>
<td>Block Coverage</td>
<td>263</td>
<td>66</td>
<td>306</td>
<td>76</td>
<td>263</td>
<td>66</td>
<td>255</td>
</tr>
<tr>
<td>Function Coverage</td>
<td>65</td>
<td>97</td>
<td>66</td>
<td>99</td>
<td>65</td>
<td>97</td>
<td>64</td>
</tr>
<tr>
<td>Faults Detected</td>
<td>22</td>
<td>54</td>
<td>24</td>
<td>58</td>
<td>23</td>
<td>56</td>
<td>23</td>
</tr>
</tbody>
</table>

### Technique Combination

<table>
<thead>
<tr>
<th>Combination</th>
<th>Blocks</th>
<th>Functions</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>abs</td>
<td>%</td>
<td>abs</td>
</tr>
<tr>
<td>$(WB-2 \cap US-3)$</td>
<td>273</td>
<td>68</td>
<td>65</td>
</tr>
<tr>
<td>$(WB-2 - US-3)$</td>
<td>32</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>$(US-3 - WB-2)$</td>
<td>14</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>$(WB-2 \cup US-3)$</td>
<td>319</td>
<td>80</td>
<td>66</td>
</tr>
</tbody>
</table>

### Faults ranked in tiers

<table>
<thead>
<tr>
<th></th>
<th>1-5</th>
<th>6-10</th>
<th>11-15</th>
<th>16-20</th>
<th>Rest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average sessions affected</td>
<td>98% (83)</td>
<td>81% (69)</td>
<td>33% (28)</td>
<td>1% (1)</td>
<td>0% (0)</td>
</tr>
<tr>
<td>WB-1 detects</td>
<td>100% (5)</td>
<td>80% (4)</td>
<td>60% (3)</td>
<td>40% (2)</td>
<td>40% (16)</td>
</tr>
<tr>
<td>WB-2 detects</td>
<td>100% (5)</td>
<td>60% (3)</td>
<td>40% (2)</td>
<td>40% (2)</td>
<td>50% (20)</td>
</tr>
</tbody>
</table>
Threats to Validity

• Need to study additional websites
• Experiment users may not be representative of normal users
• No comparison to other white box techniques
• Tester may not have implemented white box testing properly
• Fault seeding may be biased
• Uneven test suite size between techniques
Additional Considerations

• Web-Application State
  – Output depends on more than URL, name-value pairs
  – Test cases have different meaning in different states

• Non-Determinism
  – Identical sets of input can produce different outputs

• Managing Evolving Test Suites
  – Over time large number of user sessions accumulated
  – Remove redundant cases my keeping same function, page, block coverage
Conclusion

• Pros
  – New technique for web-testing
  – Appears to complement existing techniques
  – Not dependent on underlying technology
  – Little human effort required

• Cons
  – More experimentation needed
  – Requires stable application...good for beta testing
  – Practicality?
Future Work

• Combining traditional techniques with user-session tests
• Filtering and reducing large amounts of user-sessions
• Costs of this technique versus others
Questions