Learning Effective Oracle Comparator Combinations for Web Applications

Sara Sprenkle

Emily Hill
Lori Pollock
Testing Web Applications

- 772 million people online in May 2007*
- Amazon.com had 142K unique visitors, on average visit 3.1 times, in May 2007*
  - Earned $10.7 billion in revenue in 2006

Need reliable web applications

- Errors are seen by many, quickly
- Frequent maintenance/update cycles
- Need fast, effective oracles

* Source: comScore
Why Oracle Effectiveness Matters

- Need oracle to be accurate
  - Allow only the applications that behave as expected to be deployed
- False negatives (say OK when faulty)
  - Loss of revenues, consumer confidence
- False positives (say faulty when not)
  - Overwhelm testers, waste developers’ time
- Testers need to consider tradeoffs between false negatives, false positives
  - May use >1 oracle to improve effectiveness
Oracle Selection Problem

Given a suite of oracle comparators, which oracle comparator or combination should you select for your web application?

• Most effective oracle combination
• Select cheaply, systematically
Overview of Our Approach

- **Decision tree learning** to determine systematically best oracle comparator
  - Intuitive, easily interpreted model
- Train on oracles’ failure detection results for seeded faults, application behavior
Overview of Contributions

- **Learned** effective oracle combinations for 4 representative web applications using decision tree learning
- **Evaluated** learned oracles on non-training set data
- **Proposed** a *methodology* for selecting an effective oracle comparator combination
Background: Oracle Comparators

- Suite of 22 general, automated HTML oracle comparators [ISSRE07]
  - HTML: common output format
  - What user sees
  - Failures from other output may show up in HTML
Example of an HTML Response

```html
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>hiperspace lab</title>
<style>
  a:hover{color:#952C2C; text-decoration:none}
</style>
<script language="Javascript"> ...
</script>
</head>
<body>
<table border=0 cellspacing=0 width="580">
  <tr><td rowspace=2>
    <img src="books.gif" width=310></td></tr>
  ...
</table>
<!-- Sidebar Links -->
<ul>
  ...
  <li><a href="alumni.html">Alumni</a>
</ul>
</body>
</html>
```

- Start tag
- Content
- Style
- Layout
- Comment
- Content
- Attribute
- Close tag

Browser collapses white space
HTML Oracle Challenges

- Many different HTML components
- Differences in various components may or may not matter
  - Which matter depends on the application
- Created 22 customized oracles to address these differences
  - Tradeoffs between oracles’ false positives and false negatives
**Background: Our Oracle Process**

- **Expected results:** use original version of application (assumed to be correct)
  - **Gold Standard**

---

**Diagram Notes:**
- **Input:** HTTP Request
- **Output:** Expected HTML Response, Actual HTML Response
- **Web App Under Test:**
  - Gold Standard
  - Expected HTML Response
  - Actual HTML Response
Background: Our Oracle Process

Comparator

- Expected HTML Response
- Actual HTML Response
- Customized HTML Processor
- Processed Expected HTML Response
- Processed Actual HTML Response

Pass?

- Pass/No Pass
- Set of Differences
Partial Ordering of Implemented Comparison Algorithms

**Document-based oracles:**
- Document (D)
- DocumentBase (DB)
- DocumentBase-CollapsedWS (DB-W)

**Content-based oracles:**
- Content (C)
- Content-CollapsedWS (C-W)
- Content-CollapsedWS-Dates (C-WD)

**Structure-based classes:**
- Tags (T)
  - Forms (N+F)
  - TagNames+Impt Attrs (N+I)
    - TagNames (N)
    - UnorderedLinks (N+U)

More information → Fewer False Negatives → Fewer False Positives

Sprenkle • STEV2007
ISSRE07 Combinations Summary

- Investigated if unioning or intersecting oracle comparators → a more effective oracle
  - *Unioning* carefully selected comparators: better effectiveness than individual comparators
- Across all apps, \( N+I \cup C-WD \) was best
- Did not exhaustively combine oracles or try different operations to combine

![Diagram showing combinations of oracles]

- Oracle 1 + Oracle 2
- Oracle 1 + Oracle 3
- Oracle 1 + Oracle 4
- ...
Selecting an Effective Oracle

- Given the tradeoffs between all the classes and combinations, need **systematic** technique to select **effective** oracle

- Decision Tree Learning
  - Inductive learning method
  - Constructs a classifier/decision tree for a training data set

- Our classification problem:
  - Given: a feature set containing oracle results, application behavior
  - Classify an HTTP request/HTML response as pass/no pass
Decision Tree Learning

**Training Data**

- Each oracle’s actual pass/no pass results
- Application Behavior
- Human-generated expected oracle results

Diagram:

- Input: Training Data (Decision Tree Learner (C5.0))
- Output: Decision Tree
  - Oracle Combination
Example of Training Data

<table>
<thead>
<tr>
<th>Request/Response</th>
<th>App. Behavior</th>
<th>Response Behavior</th>
<th>Oracle 1</th>
<th>...</th>
<th>Oracle n</th>
<th>Expected Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D</td>
<td>D</td>
<td>P</td>
<td></td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
<td>D</td>
<td>P</td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td></td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>N</td>
<td>D</td>
<td>P</td>
<td></td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>N</td>
<td>D</td>
<td>N</td>
<td></td>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>
Methodology: Generating Training Data

Faults seeded manually into application code

Various categories of seeded faults
  - data, logic, form, appearance, link
Methodology: Generating Training Data

For the test suite of each subject application:

- Compare the HTML document from faulty version with HTML document from clean version
- Manually determine if document exposes fault

Test Suite  ➞  Replay suite on clean and fault-seeded version(s) of code  ➞  Generate failure detection reports using each oracle
Subject Applications

Four deployed subject applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Test Cases</th>
<th>Requests</th>
<th>NCLOC*</th>
<th>Faults Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masplas</td>
<td>169</td>
<td>1103</td>
<td>999</td>
<td>22</td>
</tr>
<tr>
<td>E-Commerce Bookstore (Book)</td>
<td>125</td>
<td>3564</td>
<td>7791</td>
<td>36</td>
</tr>
<tr>
<td>Course Project Manager (CPM)</td>
<td>890</td>
<td>12352</td>
<td>9300</td>
<td>96</td>
</tr>
<tr>
<td>DSpace</td>
<td>75</td>
<td>3183 (3023 HTML pages)</td>
<td>49513</td>
<td>20</td>
</tr>
</tbody>
</table>

*Non-Commented Lines of Code

Total: 174 Faults

*Non-Commented Lines of Code
## Expected Oracle Results

Humans determined if an HTML response exposed a failure

<table>
<thead>
<tr>
<th>App</th>
<th>Pass (Trivial)</th>
<th>Pass (Non-Trivial)</th>
<th>No Pass</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masplas</td>
<td>22924</td>
<td>1208</td>
<td>134</td>
<td>24266</td>
</tr>
<tr>
<td>Book</td>
<td>107990</td>
<td>18764</td>
<td>1550</td>
<td>128304</td>
</tr>
<tr>
<td>CPM</td>
<td>1164661</td>
<td>19449</td>
<td>1682</td>
<td>1185792</td>
</tr>
<tr>
<td>DSpace</td>
<td>53356</td>
<td>5651</td>
<td>4653</td>
<td>63660</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1348931</strong></td>
<td><strong>45072</strong></td>
<td><strong>8012</strong></td>
<td><strong>1402022</strong></td>
</tr>
</tbody>
</table>
Decision Tree Learning

Training Data
- Each oracle’s actual pass/no pass results
- Application Behavior
- Human-generated expected oracle results

Misclassification costs:
Assign costs to false +, false -, total error

For each application, learned decision tree 3 times:
- Reduce false +, false -, total error
Example Decision Tree Output

DSpace - Reduce False Negatives

Decision tree:

cwd = 1: 1 (6164/522)
cwd = 0:
  ....f = 0: 0 (57487)
  f = 1: 1 (9)

Says “no pass”

C-WD \cup N+F

Says “no pass”

# right/wrong

Says “pass”
Measuring Effectiveness

- Do not include trivial passes
  - Better distinguish between comparators

- False positive error rate
  \[
  \frac{\# \text{ reported failures}}{\# \text{ non-trivial pass} + \# \text{ no pass}}
  \]

- False negative error rate
  \[
  \frac{\# \text{ failures missed}}{\# \text{ non-trivial pass} + \# \text{ no pass}}
  \]
## Learned Oracle Results

<table>
<thead>
<tr>
<th>App</th>
<th>Oracle</th>
<th>False -</th>
<th>False +</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masplas</td>
<td>DB-W</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Book</td>
<td>T ∪ C</td>
<td>4.25</td>
<td>0</td>
<td>4.25</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0</td>
<td>7.63</td>
<td>7.63</td>
</tr>
<tr>
<td>CPM</td>
<td>dD ∪ N+F-Sel</td>
<td>0.43</td>
<td>0</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0</td>
<td>7.96</td>
<td>7.96</td>
</tr>
<tr>
<td>DSpace</td>
<td>N+U</td>
<td>.45</td>
<td>0.04</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>N-S+U</td>
<td>0.82</td>
<td>0</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>C-WD ∪ N+F</td>
<td>0</td>
<td>5.07</td>
<td>5.07</td>
</tr>
</tbody>
</table>

- Generally, low error rates
- Tradeoffs between false positives/false negatives
# Learned Oracle Results

<table>
<thead>
<tr>
<th>App</th>
<th>Oracle</th>
<th>False -</th>
<th>False +</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masplas</td>
<td>DB-W</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Book</td>
<td>T ∪ C</td>
<td>4.25</td>
<td>0</td>
<td>4.25</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0</td>
<td>7.63</td>
<td>7.63</td>
</tr>
<tr>
<td>CPM</td>
<td>dD ∪ N+F-Sel</td>
<td>0.43</td>
<td>0</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>0</td>
<td>7.96</td>
<td>7.96</td>
</tr>
<tr>
<td>DSpace</td>
<td>N+U</td>
<td>.45</td>
<td>0.04</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>N-S+U</td>
<td>0.82</td>
<td>0</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>C-WD ∪ N+F</td>
<td>0</td>
<td>5.07</td>
<td>5.07</td>
</tr>
</tbody>
</table>

- Cross-validation: applying learned oracle on other three applications did not yield an effective oracle.
Evaluation of Learned Oracles

- Evaluate on non-training set data
- Experiment 1: Effect of nondeterministic, real-time behavior
  - Focus on false positives
  - Executed CPM and DSpace test suite 9 times over several months
- Experiment 2: Failure detection
  - Seeded faults in all four applications
  - Two sets of data
    - Failure detection results
    - Normalized failure detection results to reduce bias towards CPM
Experiment 1 Results

Learned from CPM and DSpace

- Higher error because compared test suite executions separated by several months

<table>
<thead>
<tr>
<th>Oracle Comparator</th>
<th>False Positive Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPM</td>
</tr>
<tr>
<td>N+U</td>
<td>0</td>
</tr>
<tr>
<td>N-S+U</td>
<td>0</td>
</tr>
<tr>
<td>dD ∪ N+F-Sel</td>
<td>38.7</td>
</tr>
<tr>
<td>C-WD ∪ N+F</td>
<td>98.8</td>
</tr>
<tr>
<td>N+I ∪ C-WD</td>
<td>100</td>
</tr>
<tr>
<td>T ∪ C</td>
<td>100</td>
</tr>
<tr>
<td>DB-W</td>
<td>100</td>
</tr>
<tr>
<td>D</td>
<td>100</td>
</tr>
</tbody>
</table>
Experiment 2 Results

- Error rates < 1.5% if include trivial pass
- Fewest false negatives: document-based
- Need to consider both structure, content
Methodology for Selecting an Oracle Comparator Combination

1. Create oracle training data
   - Use faults from bug reports
   - Execute oracles on test-suite responses
   - Manually determine pass/no pass results
   - Identify responses with nondeterministic behavior

2. Use decision trees to learn the best oracle combination
   - Tailor to goals: fewest false positives, false negatives, total error

3. Evolve combination as application evolves
   - Application’s behavior and what constitutes a failure may change
   - Repeat steps 1 and 2
Related Work

- HTML-based oracle comparators
  - Elbaum05, DiLucca 02 -- few details of implementation, no evaluation of false positives, negatives with nondeterminism

- Machine learning to classify program executions as pass/no pass
  - Bowring04, Haran07, Podgurski03
  - Use execution profile for features
Conclusions & Future Work

- Proposed methodology for selecting effective oracle comparator combinations
  - Decision tree learning on representative faults from bug reports
- Evaluated learned oracles on non-training set data

Future Work
- Evaluate on additional faults
- Evaluate on additional applications