WEBVIZOR: A Visualization Tool for Applying Automated Oracles and Analyzing Test Results of Web Applications

Sara Sprenkle
Lori Pollock
Holly Esquivel*
Barbara Hazelwood*

* Supported by the CRA-W Distributed Mentor Program
World Runs on Web Applications

Google • Amazon • Travelocity • Yahoo! Mail • Facebook • eBay • Myspace.com • CNN.com • YouTube • Wikipedia • Slashdot
Challenges for Testing Web Applications

- Large, Frequently Updated Code Base
- Large Test Suite
- Various output formats
- Oracles: manual or automated; false positives & false negatives
Analyzing Test Suite Results

Expected Structure

Differences Between Structure

Actual Structure
Existing Tools

- JUnit-based tools: Cactus, WebTest, HttpUnit, HtmlUnit, etc.
  - Expect that testers do unit testing
  - Require manually writing oracles for each test case
  - Can’t visualize the responses

- Selenium
  - Firefox-based capture/replay tool
  - Visualize results, navigate through a test case
  - Can’t compare actual and expected results visually
  - Limits test cases
  - Requires manually writing oracles for each test case

Sprenkle • TAICPART 2008
The Problem

➢ No existing automated tool that allows testers to
  • View actual and expected results, together
  • Apply automated oracle comparators
    ➢ See symptoms of failures
  • View results in context of test case
  • Navigate through test results
Our Tool: **WEBVIZOR**

- **WEB** Application Failure Detection **V**isualization With **O**racles

- **Features:** Does all that… and MORE!
  - Includes set of oracle comparators from Sprenkle et al. [ISSRE 2007]
  - Plug-in new oracle comparators
  - Oracles’ views of actual, expected results

- **Impact**
  - Easier to identify failures
  - Eliminate human tester’s manual analysis
  - Direct development of new oracles
Background: Our Oracle Process

One way to generate expected results: use version of application that is assumed to be correct
Automated Oracle Comparators

➢ Implemented 22 oracle comparators
  • Focused on different parts of HTML

➢ Experimental results:
  • Tradeoffs in false positives, false negatives
  • No ideal comparator for all applications
  • Sometimes better effectiveness of use combinations of comparators
Testing With **WEBVIZOR**

**WEBVIZOR**

- Pluggable Oracles
- Set of Differences
- Processed Responses
- Pass/No Pass

Customize which oracles
Implementation

➢ Runs as a web application on local web application server
  • Leverage browser’s HTML rendering capability

➢ Technologies
  • Frontend: Java Servlets, JSPs, JavaScript
  • Backend: Java; Oracles written in Java, Perl

➢ Goal: Portable
  • Tested on Linux & Mac using Resin and Tomcat web application servers
  • Limitation: File system dependence
Tour of WEBVIZOR using DSpace

- Customized digital publications
  - Generates sorted publications
  - Research group maintains publications database
  - Implemented by an undergraduate

- Verify modification to the code

Submit: 1

2008


2007


Sprenkle • TAICPART 2008
Initializing **WebVizOr**

Comparison View Input

Single View Input

Running on local machine
View and Visually Compare Test Results

Test case: View

View/Oracle Selection Buttons

Navigation Buttons

Sort Criteria: Date Title Type First Author Project
Search For Author: None Search

2005
Sort by: Title Type Project


View and Visually Compare Test Results

Several Oracles Detect Differences
View Oracle Comparator Results

HTML Source view
View Oracle Comparator Results
View and Visually Compare Test Results
View and Visually Compare Test Results
View Oracle Comparator Results
View Oracle Comparator Results

Test Suite Results: 0019.tc : 0002.1 : Text Content

54 Putting Escape Analysis to Work for Software Testing
55 Inter-class Def-Use Analysis with Partial Class Representations
56 Incremental Call Graph Reanalysis for Object-Oriented Software Maintenance
57 Using Path Spectra to Direct Function Cloning
58 Demand-driven Inlining Heuristics in Region-based Optimization for ILP Architectures
59 &nbsp;
60 DSpace Software
61 Copyright &nbsp; &copy; &nbsp; 2002-2004 &nbsp;
62 MIT
63 &nbsp; and &nbsp;
64 Hewlett-Packard
65 &nbsp;-
66
67 Feedback

53 Recent &nbsp; Submissions
54 Inter-class Def-Use Analysis with Partial Class Representations
55 Putting Escape Analysis to Work for Software Testing
56 Incremental Call Graph Reanalysis for Object-Oriented Software Maintenance
57 Using Path Spectra to Direct Function Cloning
58 Demand-driven Inlining Heuristics in Region-based Optimization for ILP Architectures
59 &nbsp;
60 DSpace Software
61 Copyright &nbsp; &copy; &nbsp; 2002-2004 &nbsp;
62 MIT
63 &nbsp; and &nbsp;
64 Hewlett-Packard
65 &nbsp;-
66
67 Feedback
Not a failure: Requirements do not specify an order
Navigate through Test Results

Collapsible Navigation Panel

Navigation Buttons
Automated Replay and Failure Detection for Web Applications

Sara Srenkle, Emily Gibson, Sreedevi Sampath, and Lori Pollock

User-session-based testing of web applications gathers user sessions to create and continually update test suites based on real user input in the field. To support this approach during maintenance and beta testing phases, we have built an automated framework for testing web-based software that focuses on scalability and evolving the test suite automatically as the application's operational profile changes. This paper reports on the automation of the replay and oracle components for web applications, which pose issues beyond those in the equivalent testing steps for traditional, stand-alone applications. Concurrency, non-determinism, dependence on persistent state and previous user sessions, a complex application infrastructure, and a large number of output formats necessitate developing different replay and oracle comparator operators, which have tradeoffs in fault detection effectiveness, precision of analysis, and efficiency. We have designed, implemented, and evaluated a set of automated replay techniques and oracle comparators for user-session-based testing of web applications. This paper describes the issues, algorithms, heuristics, and an experimental case study with user sessions for two web applications. From our results, we...
ISSRE 2007 Experimental Study

- Goal: Evaluate accuracy of 22 automated oracle comparators
  - Reported failures, false positives, false negatives
- Seeded 252 faults into 4 subjects applications
- Executed test suite on each fault-seeded version of application
- Analyzed 10K+ responses to determine if response contained symptom of seeded fault
- Required time: a few days
  - Manually? I don’t want to think about it
Experience Using **WebVizOR**

 dévelop (1) Develop new oracles
• Analyzed test results using Sprenkle et al.’s four oracle comparators [ASE2005]
• Identified common situations of oracles reporting false positive or false negative
  ➢ Recognized similar oracle behavior more easily with **WebVizOR**
Demonstration
WEBVIZOR Features Summary

➤ Viewing and visually comparing results

➤ Oracle integration
  • Automatic execution of oracles, on demand
  • Visualize oracle results
  • Plug-in new oracles

➤ Ease of navigation
  • Iterate through responses, test cases
  • Response, test case navigation panels

➤ View detailed requests in context of results
Conclusions

➢ **WEBVIZOR** allows
  • Analyzing test results more easily
  • Plugging in new oracles
  • Development of new oracles

➢ Future work
  • Test on Windows
  • Add features to improve failure detection visualization and usability

➢ Web page: [http://www.cis.udel.edu/~hiper/webvizor](http://www.cis.udel.edu/~hiper/webvizor)