A Case Study of Automatically Creating Test Suites from Web Application Field Data

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

UNIVERSITY OF DELAWARE

Evolving Web Applications

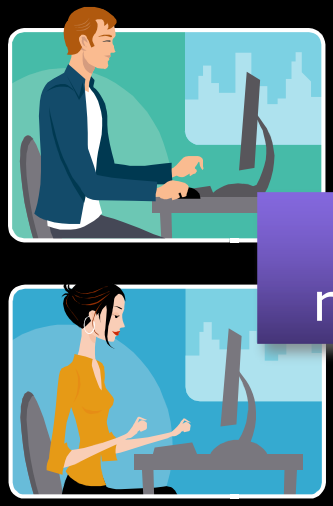
- Code constantly changing
  - Fix bugs
  - Add functionality
  - Improve performance
- Global user base, available 24/7
  - Users increasingly dependent on functionality
  - Partial failures cost millions *per hour*[*]
- Motivates
  - Continuous testing of web apps as they evolve
  - Using field data to test frequently accessed code

* Michal Blumenstyk. "Web Application Development - Bridging the Gap between QA and Development." http://www.stickyminds.com, 2002.

Capture/Replay Testing

- Advantages:
  - Reproduce failures from user input
  - Prioritize bug fixes
  - Verify configuration and code upgrades
  - Complementary to other testing techniques
- Web application benefits:
  - Cheap (compared to other domains)
  - Portable (independent of underlying technology)

Capture/Replay Testing of Web Applications



User requests

Phase I: Capture

Deployed Web Application **v 0.9**

Logger

Logged requests

Captured Log

Parse Log

Phase II: Replay

Test Cases

Replay

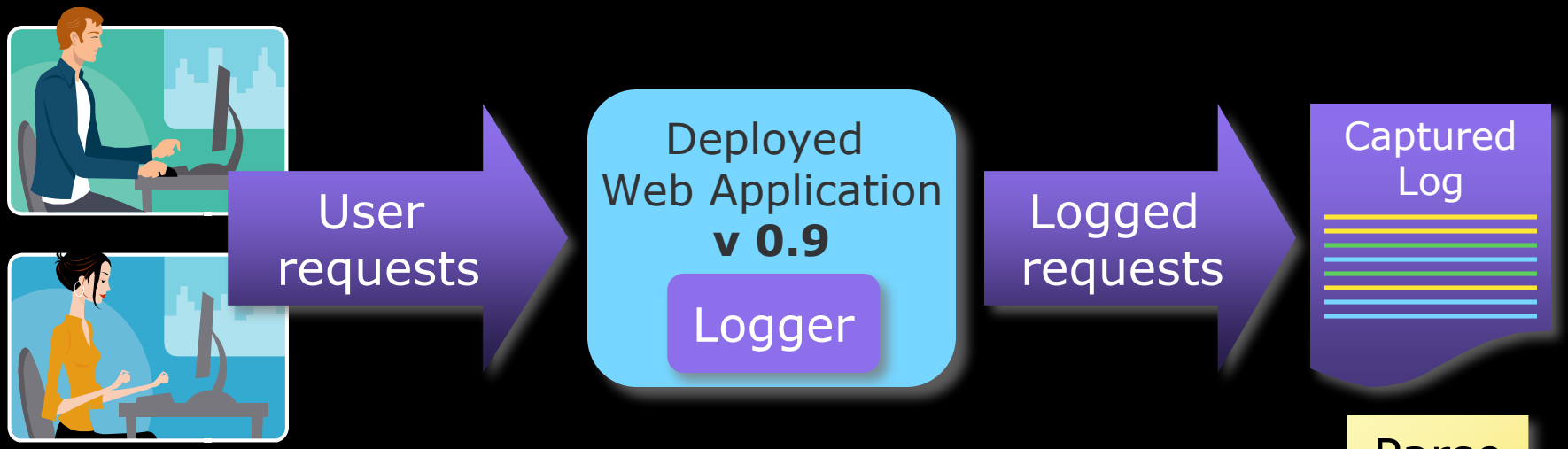
Web Application Under Test **v 1.0**

Oracle

Error Report

Existing Approach: User-session-based Testing

A specific type of capture/replay testing



- **User session**: a single user's requests
  - Identified by cookie or IP address
  - Replayed sequentially
  - Ordered by first request
- Elbaum, et al. '03: User sessions nearly as effective as model-based test cases

User-session-based Testing

- Advantages:
  - Ease debugging (replay only one user's requests)
  - Maintain a single user's state during replay
- Limitation: *lose multi-user interactions*
  - Doesn't emulate deployed behavior
  - Will miss bugs caused by user interactions
    - Users affect shared application state & change behavior of other users

Example Limitation

Bookstore Application

- User1: buys *Pure Drivel*
- User2: admin adds *Pure Drivel* to DB

Captured Log		**Deployed Behavior**
User	Request	Response
User1	index.jsp	Enter bookstore site
User2	addBook.jsp	Add book *Pure Drivel* to DB
User1	search.jsp	List of Steve Martin's books
User1	buy.jsp	Buy the book *Pure Drivel*

Example Limitation: Losing Multi-user Interaction

Captured Log		Deployed Behavior
User	Request	Response
User1	index.jsp	Enter bookstore site
User2	addBook.jsp	Add book *Pure Drivel* to DB
User1	search.jsp	List of Steve Martin's books
User1	buy.jsp	Buy the book *Pure Drivel*

Replayed User Sessions		Replayed Behavior
User	Request	Response
User1	index.jsp	Enter bookstore site
User1	search.jsp	List of Steve Martin's books (no *Pure Drivel)*
User1	buy.jsp	**ERROR:** Try to buy *Pure Drivel* (no such book in DB)
User2	addBook.jsp	Add book *Pure Drivel* to DB

Example Limitation: Losing Multi-user Interaction

Captured Log		Deployed Behavior
User	Request	Response
User1	index.jsp	Enter bookstore site
User2	addBook.jsp	Add book *Pure Drivel* to DB
User1	search.jsp	List of Steve Martin's books
User1	buy.jsp	Buy the book *Pure Drivel*

Replayed User Sessions		Replayed Behavior
User	Request	Response
User1	index.jsp	Enter bookstore site
User1	search.jsp	List of Steve Martin's books (no *Pure Drivel)*
User1	buy.jsp	**ERROR:** Try to buy *Pure Drivel* (no such book in DB)
User2	addBook.jsp	Add book *Pure Drivel* to DB

Alternative: Replay Captured Log

- Advantage
  - Replayed behavior ≈ deployed behavior
- Disadvantages
  - Difficult Debugging
    - Must replay whole log
    - Multiple users interacting
  - Large log → replay repetitive requests

- Research Focus: generate test cases to address disadvantages of both approaches

Research Contribution

- **Key Insight:** to emulate deployed behavior, test cases based on field data must not ignore multi-user interactions
- **Contribution:** 3 alternative test case generation strategies using field data
  - Better emulate deployed behavior
  - Expose different application behaviors from user sessions

Test Case Generation Strategies

- Objectives
  - Maintain multi-user interactions
  - Maintain logical user sessions
  - Low execution overhead
  - Effectiveness
    - Program coverage
    - Fault detection (future)
- Proposed Strategies
  - Fixed-Time Block
  - Server Inactivity
  - Augmented User Sessions

Fixed-Time Block

Captured Log	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

Interval=10 min	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

- Baseline (simplest)
- Test cases = activity snapshot w.r.t. server (not user)
- Short interval, more test cases
- Long interval, less test cases
- Split many logical user sessions

Fixed-Time Block

Captured Log	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

Interval=10 min	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

- Baseline (simplest)
- Test cases = activity snapshot w.r.t. server (not user)
- Short interval, more test cases
- Long interval, less test cases
- Split many logical user sessions

Server Inactivity

Captured Log	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

Threshold=4 min	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

- Smart fixed-time
- Test cases = activity snapshot w.r.t. server (not user)
- Short threshold, split more logical user sessions
- Long threshold, aggregate more logical user sessions
- Split fewer logical user sessions

Server Inactivity vs. Fixed-Time Block

Captured Log	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

Server Inactivity Threshold=4 min	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

Fixed-Time Block Interval=10 min	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

User Sessions

Captured Log	
00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

User Sessions	
00:00	User 1
00:02	User 1
00:04	User 1
00:30	User 1
00:03	User 2
00:09	User 2
00:05	User 3
00:22	User 3
00:23	User 3
00:31	User 3
00:18	User 4
00:29	User 4

- Logical user sessions
- Lose multi-user interaction
- Test cases = activity snapshot w.r.t. user

Augmented User Sessions

- Logical user sessions
- Multi-user interaction

- Test cases = activity snapshot w.r.t. server *and* user
- Does not partition log → many redundant requests

3x increase!

Captured Log

00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

00:00	User 1
00:02	User 1
00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1

00:03	User 2
00:04	User 1
00:05	User 3
00:09	User 2

00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4

00:05	User 3
00:09	User 2
00:18	User 4
00:22	User 3
00:23	User 3
00:29	User 4
00:30	User 1
00:31	User 3

Summary of Proposed Strategies

Approach	Advantages	Limitations
User Sessions	• Logical user sessions	• No multi-user interaction
Fixed-Time Block	• Multi-user interaction • Control size & number of test cases in suite	• Requires smart interval • May split logical user sessions
Server Inactivity	• Multi-user interaction • Control size & number of test cases in suite	• Requires smart threshold • May split logical user sessions
Augmented User Sessions	• Logical user sessions • Multi-user interaction	• Large test cases • Redundant requests • Higher generation cost

Case Study

- Research Question: How do the proposed strategies compare to user sessions as test cases?
- Subject Application: DSpace
  - Customized digital publications library
  - Written in Java Servlets, JSPs
  - PostGreSQL and filestore backends
  - Collected field data from Aug ’05 - Feb ’06

Classes	Methods	NCLOC	Statements	Distinct URLs	Total URLs
355	1,534	61,720	27,136	443	16,275

Case Study Methodology

- Test case generation strategies
  - User Sessions
  - Fixed-Time Block (minute, hour, & 6-hour intervals)
  - Server Inactivity (25 min threshold)
  - Augmented User Sessions
- Measures
  - Program coverage
  - Generation cost
  - Replay cost

Results:
Program Statements Covered

User Sessions	Fixed-Time Block			Server Inactivity	Augmented User Sessions
	Minute	Hour	6-hour		
17,536	12,270	15,713	17,674	17,745	17,866

- Suites cover ~65% of the code
  - Non-covered code: app setup before logging, admin, redundant classes from maintenance
- Minute & Hour split logical user sessions
  - Execute error code (redundantly)

Coverage Comparison: User Sessions (US) vs. Alternative Strategies (A)

US coverage = 17,536

Suite (A)	A	US ∪ A	US - A	A - US
6-hour	17,674	17,731	74	212
Server Inactivity	17,745	17,947	219	428
Augmented User Sessions	17,866	17,971	156	452

- US ∪ A: total coverage if run both suites
- US - A: what US covers & Alternative misses
- A - US: what Alternative covers & US misses

Results:
Generation & Replay Costs

Suite	No. Test Cases	Generation Time	No. URLs	Replay Time
User Sessions	1,342	2 s	16,275	76 min
Minute	8,447	2 s	16,275	216 min
Hour	1,769	3 s	16,275	102 min
6-hour	508	4 s	16,275	73 min
Server Inactivity	1,814	2 s	16,275	75 min
Augmented US	1,342	8 s	184,656	N/A
Captured Log	1	3 s	16,275	52 min

- Augmented US - significantly redundant no. requests
- Server Inactivity comparable to US
- Why not just replay captured log?
  - Single test case, can't reduce
  - Difficult to debug

Results:
Generation & Replay Costs

Suite	No. Test Cases	Generation Time	No. URLs	Replay Time
User Sessions	1,342	2 s	16,275	76 min
Minute	8,447	2 s	16,275	216 min
Hour	1,769	3 s	16,275	102 min
6-hour	508	4 s	16,275	73 min
Server Inactivity	1,814	2 s	16,275	75 min
Augmented US	1,342	8 s	184,656	N/A
Captured Log	1	3 s	16,275	52 min

- Augmented US - significantly redundant
- Server Inactivity comparable to US
- Why not just replay captured log?
  - Single test case, can't reduce
  - Difficult to debug

Results:
Generation & Replay Costs

Suite	No. Test Cases	Generation Time	No. URLs	Replay Time
User Sessions	1,342	2 s	16,275	76 min
Minute	8,447	2 s	16,275	216 min
Hour	1,769	3 s	16,275	102 min
6-hour	508	4 s	16,275	73 min
Server Inactivity	1,814	2 s	16,275	75 min
Augmented US	1,342	8 s	184,656	N/A
Captured Log	1	3 s	16,275	52 min

- Augmented US - significantly redundant
- Server Inactivity comparable to US
- Why not just replay captured log?
  - Single test case, can't reduce
  - Difficult to debug

Results:
Generation & Replay Costs

Suite	No. Test Cases	Generation Time	No. URLs	Replay Time
User Sessions	1,342	2 s	16,275	76 min
Minute	8,447	2 s	16,275	216 min
Hour	1,769	3 s	16,275	102 min
6-hour	508	4 s	16,275	73 min
Server Inactivity	1,814	2 s	16,275	75 min
Augmented US	1,342	8 s	184,656	N/A
Captured Log	1	3 s	16,275	52 min

- Augmented US - significantly redundant
- Server Inactivity comparable to US
- Why not just replay captured log?
  - Single test case, can't reduce
  - Difficult to debug

Observations

- Same requests → different app behaviors
- Augmented User Sessions best emulate deployed behavior
- To maximize DSpace coverage, replay US + Augmented US
- Multi-user test cases revealed problem in Dspace's text search engine (Lucene)
  - User sessions did not find
  - Already known, fixed in later versions of DSpace

Future Work

- Larger empirical study
  - More apps, larger captured logs
  - Evaluate fault detection effectiveness
  - Compare reduced suites
  - Evaluate ease of debugging
- Recommendations to testers
  - Which test cases appropriate for different testing goals?

Contributions

- User-session-based testing limitations
  - Lose multi-user interactions
  - Ignore multi-user state dependences
- Proposed test case generation strategies
  - Using field data
  - Maintain multi-user interactions & state dependences
- Case study results
  - Proposed strategies comparable in cost & effectiveness to User Sessions
  - Augmented User Sessions and Server Inactivity most effective in terms of program coverage

Contributions

- User-session-based testing limitations
  - Lose multi-user interactions
  - Ignore multi-user state dependences
- Proposed test case generation strategies
  - Using field data
  - Maintain multi-user interactions & state dependences
- Case study results
  - Proposed strategies comparable in cost & effectiveness to User Sessions
  - Augmented User Sessions and Server Inactivity most effective in terms of program coverage