Prioritizing User-session-based Test Cases For Web Application Testing

Sreedevi Sampath[1], Renee Bryce[2],
Gokul Viswanath[1], Vani Kandimalla[2], A. Gunes Koru[1]

[1] UMBC

[2] UNLV
UNIVERSITY OF NEVADA LAS VEGAS

April 10, 2008

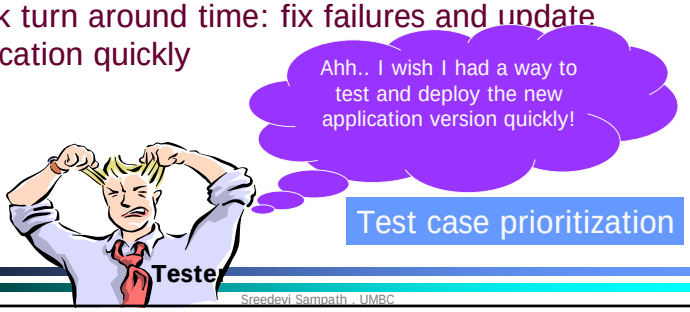
Need for Reliable Web Applications

- At the end of 2007, 1.3 billion users were online worldwide[1]
- 2007 US retail holiday e-commerce spending surpasses 29.2 billion dollars. Up 19% from last year.[2]
- Huge losses on web site failure: tune of millions of dollars per hour[3]
- Large number of failures during maintenance[4]

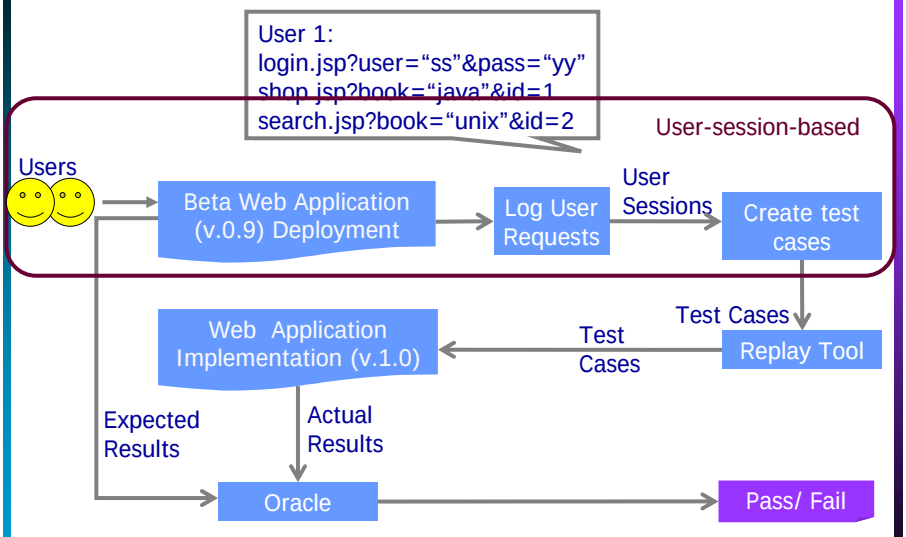
1. Internet World Stats (www.internetworldstats.com)
2. comScore (www.comscore.com)
3. Web Application Development - Bridging the Gap between QA and Development by Michal Blumenstyk
4. Causes of Failures in Web Applications by Solia Pertet and Priya Narsimhan. December 2005

Factors Complicating Web Application Testing

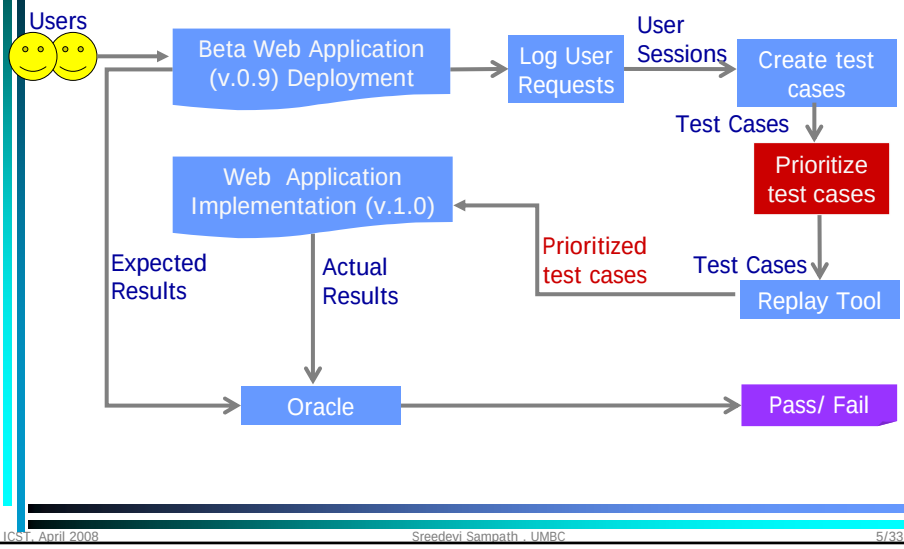
- Numerous technologies (HTML, Java, ASP, MySQL, etc.) and components
- Combination of GUIs, database, distributed and stand-alone applications
- Automatically update on server-side without user intervention—be careful about nature of update!
- High availability and visibility requirement
- Quick turn around time: fix failures and update application quickly



User-session-based Testing Process



User-session-based Test Case Prioritization



Test Case Prioritization: Traditional

- Schedule test cases according to some criterion such that they meet some performance goal
- Existing work [Rothermel et al. TSE 00, Do et al. TSE 05, Elbaum et al. TSE 01, Binkley ICSM 92, Jones et al. TSE 03, Offutt et al. ICTS 95, Bryce et al. DOSTA 07]
  - Prioritization criteria: code coverage, fault likelihood, fault exposure potential, several others
  - Performance goals: rate of fault detection, average percent of faults detected (APFD)
- Motivation for new prioritization strategies
  - Large number of user sessions used in testing
  - Quick turn-around time for web systems
  - Characteristics of web application test cases can be exploited

Our Contributions

- Define new criteria for prioritizing user-session-based test cases
  - Length-based
    - Request-Long to Short
    - Request-Short to Long
    - Parameter-value-Long to Short
    - Parameter-value-Short to Long
  - Frequency-based
    - Most-frequently accessed sequence
    - All accessed sequences
  - Parameter-value interaction-based
    - Unique parameter-value coverage
    - Parameter-value interaction coverage
  - Empirical evaluation of criteria with three subject web applications and user sessions

Our Contributions

- Define new criteria for prioritizing user-session-based test cases
  - Length-based
    - Request-Long to Short
    - Request-Short to Long
    - Parameter-value-Long to Short
    - Parameter-value-Short to Long
  - Frequency-based
    - Most-frequently accessed sequence
    - All accessed sequences
  - Parameter-value interaction-based
    - Unique parameter-value coverage
    - Parameter-value interaction coverage
  - Empirical evaluation of criteria with three subject web applications and user sessions

Length by Number of Requests

- Test cases that cover larger number of requests are ordered before test cases that cover smaller number of requests

Original Test Case

T1:
login.jsp?user="ss"&pass="yy"
shop.jsp?book="java"&id=1
search.jsp?book="unix"&id=2

Our View for Prioritization

T1:
login.jsp
shop.jsp
search.jsp

Test Case	Login.jsp	Shop.jsp	Search.jsp
T1	X	X	X
T2			X
T3	X	X	

Prioritization order
Req-L to S: T1, T3, T2
Req-S to L: T2, T3, T1

Length by Number of Parameter-Values

- Test cases that cover larger number of parameter-values (PVs) ordered before test cases that cover smaller number of PVs

Original Test Case

T1:
login.jsp?user="ss"&pass="yy"
shop.jsp?book="java"&id=1
search.jsp?book="unix"&id=2

Our View for Prioritization

T1:
user=ss, pass=yy, book=java
id=1, book=unix, id=2

Test Case	Parameter-values (PV)	#of PVs
T1	user=ss, pass=yy, book=java, id=1, book=unix, id=2	6
T2	user=pp, pass=ab	2
T3	user=ss, pass=yy, book=perl	3

Prioritization order
PV-L to S: T1, T3, T2
PV-S to L: T2, T3, T1

Our Contributions

- Define new criteria for prioritizing user-session-based test cases
  - Length-based
    - Request-Long to Short
    - Request-Short to Long
    - Parameter-value-Long to Short
    - Parameter-value-Short to Long
  - Frequency-based
    - Most-frequently accessed sequence
    - All accessed sequences
  - Parameter-value interaction-based
    - Unique parameter-value coverage
    - Parameter-value interaction coverage
  - Empirical evaluation of criteria with three subject web applications and user sessions

Frequency-based Prioritization

- Test cases that cover most frequently accessed sequence of pages are ordered before test cases that cover less frequently accessed page sequences
  - Most Frequently Accessed Sequence (MFAS)
  - All Accessed Sequences (AAS)

Most Frequently Accessed Sequence (MFAS)

- Identify most frequently accessed request sequence (of size 2)
- Order test cases in decreasing order of appearance of this sequence

Seq1 : <login.jsp, registration.jsp>

Test Case	Seq1
T1	3
T2	7
T3	9

Prioritization order: T3,T2, T1

All Accessed Sequences (AAS)

For each sequence, beginning with the most frequently accessed sequence, test cases with maximum occurrences of these sequences are ordered before other test cases

Sequence	Frequency of access in test suite	Test cases with maximum occurrences of sequence
S1	54	T4, T10 (16)
S2	23	T2, T6, T17 (8)
S3	18	T5, T9 (5)

Prioritization order: T4, T6, T9, T17, T5, T10

Our Contributions

- Define new criteria for prioritizing user-session-based test cases
  - Length-based
    - Request-Long to Short
    - Request-Short to Long
    - Parameter-value-Long to Short
    - Parameter-value-Short to Long
  - Frequency-based
    - Most-frequently accessed sequence
    - All accessed sequences
  - Parameter-value interaction-based
    - Unique parameter-value coverage
    - Parameter-value interaction coverage
  - Empirical evaluation of criteria with three subject web applications and user sessions

Unique Parameter-Value Coverage

- Select a next test that maximizes number of parameter-values not previously present in selected tests

Test Case	Parameter-values (PV)
T1	user=ss, pass=yy, book=java, id=1, book=unix, id=2
T2	user=pp, pass=ab
T3	user=ss, pass=yy, book=perl

Prioritization order
1-way: T1, T2, T3

Parameter-Value Interaction Coverage

- Select a next test that maximizes number of 2-way parameter-value interactions between pages that occur in a test case

Parameters

Log-in (on Page 1)	Member Type (on Page 2)	Discount type (on Page 3)
New member	Basic	None
Member	Silver	$10 off
		Free Ship

Values

Possible inputs to a web application with three pages

Parameter-Value Interaction Coverage

Test Case	Log-in	Member Type	Discount status
T1	New member	Basic	None
T2	Member	Silver	$10 off
T3	New Member	Basic	Free Ship

Parameters and Values accessed in test cases

Parameter-value Interactions in Test Cases

Test Case	Parameter-value Interactions
T1	New member, Basic
	New member, None
	Basic, None
T2	Member, Silver
	Member, $10 off
	Silver, $10 off
T3	New member, Basic
	Basic, Free Ship
	New Member, Free Ship

Parameter-Value Interaction Coverage

Parameters

Test Case	Log-in	Member Type	Discount status
T1	New member	Basic	None
T2	Member	Silver	$10 off
T3	New Member	Basic	Free Ship

Values

Test Case	Parameter-value Interactions
T1	New member, Basic
	New member, None
	Basic, None
T2	Member, Silver
	Member, $10 off
	Silver, $10 off
T3	New member, Basic
	Basic, Free Ship
	New Member, Free Ship

Prioritization order
2-way: T1, T2, T3

Our Contributions

- Define new criteria for prioritizing user-session-based test cases
  - Length-based
    - Request-Long to Short
    - Request-Short to Long
    - Parameter-value-Long to Short
    - Parameter-value-Short to Long
  - Frequency-based
    - Most-frequently accessed sequence
    - All accessed sequences
  - Parameter-value interaction-based
    - Unique parameter-value coverage
    - Parameter-value interaction coverage
- Empirical evaluation of criteria with three subject web applications and user sessions

Empirical Study Design

- Three Java/JSP-based subject web applications
  - Book (7K LOC, 125 user sessions, 40 faults)
  - CPM (9K LOC, 890 user sessions, 135 faults)
  - MASPLAS (1K LOC, 169 user sessions, 29 faults)
- Evaluate prioritization techniques and compare w.r.t. random test ordering
- Measure effectiveness using
  - Rate of fault detection
  - Average Percentage Faults Detected [Rothermel TSE'00]
  - Execution time

Average Percentage Faults Detected (APFD)

Source: Rothermel TSE'00

APFD measures the weighted average of the percentage of faults detected over the life of a test suite.

Let,

T be a test suite containing n test cases,

F be a set of m faults revealed by T

TF_i be the first test case in ordering T' of T which reveals fault i

Then, the APFD for test suite T' is given as

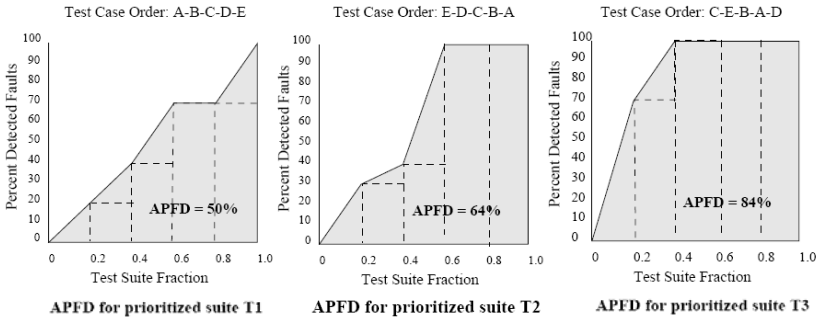
$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{nm} + \frac{1}{2n}$$

Average Percentage Faults Detected (APFD)

Source: Rothermel TSE'00

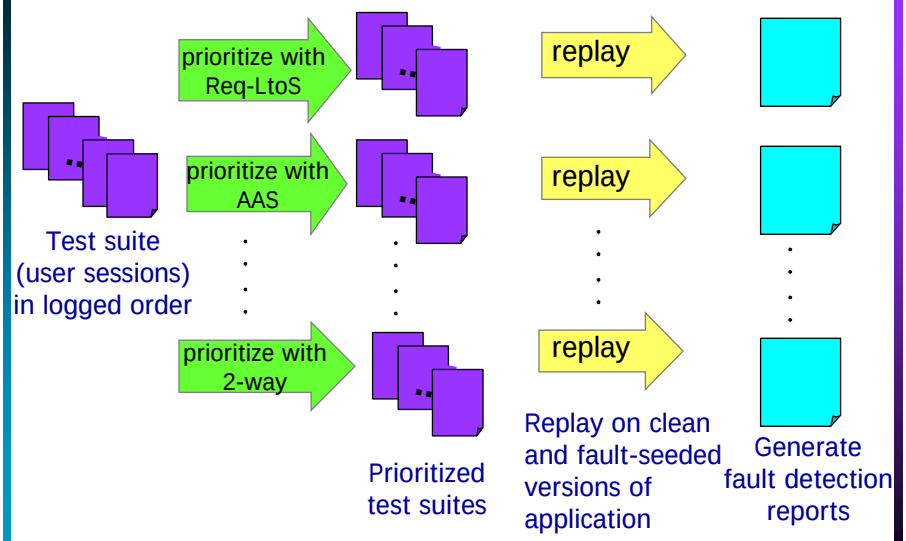
test	fault									
	1	2	3	4	5	6	7	8	9	10
A	x				x					
B						x	x			
C	x	x	x	x	x	x	x			
D					x					
E								x	x	x

Test suite and faults exposed

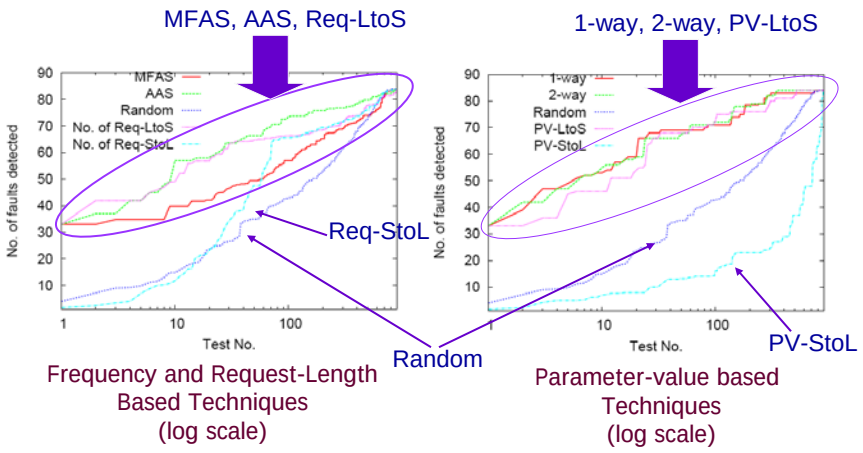


APFD for prioritized suite T1 **APFD for prioritized suite T2** **APFD for prioritized suite T3**

Methodology



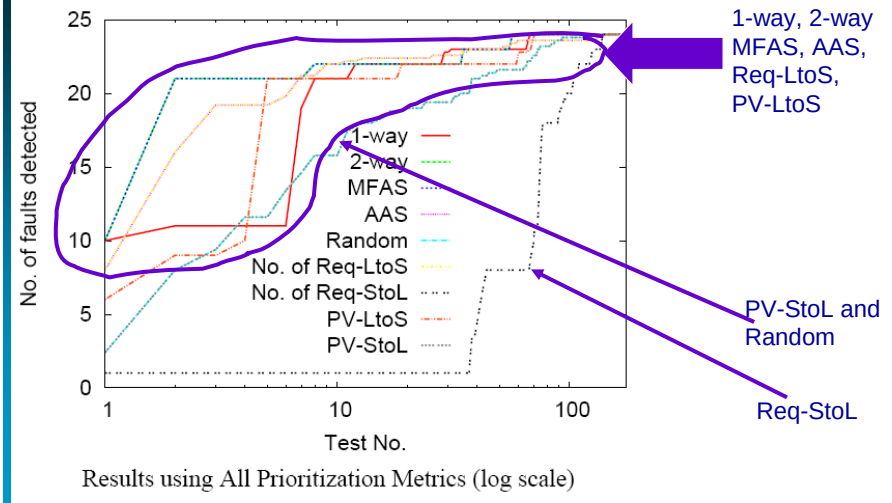
CPM: Rate of Fault Detection



CPM: Rate of Fault Detection as Measured by APFD

% of test suite run	Most Frequently Accessed request	All Accessed Sequence	No. of Requests Long to short	No. of Requests Short to long	Random	1-way	2-way	PVs Long to short	PVs Short to Long
10	65.43	**85.28**	78.17	75.14	48.63	83.79	83.72	83.53	16.38
20	74.16	88.52	80.34	77.76	57.55	87.78	**90.8**	88.77	25.6
30	77.75	89.4	81.77	80.27	64.51	91.54	**91.72**	88.77	26.44
40	79.61	89.86	84.58	81.39	69.19	94.79	**95.64**	92.71	28.76
50	80.92	91.04	85.58	82.95	73.03	94.79	**95.64**	92.71	30.33
60	81.71	91.58	87.14	84.44	75.37	94.79	**95.64**	94.26	34.64
70	82.73	92.1	87.74	85.15	77.37	94.79	**95.64**	94.26	39.15
80	84.28	92.35	88.27	86.21	78.24	94.79	**95.64**	94.26	39.58
90	84.57	92.37	88.3	86.31	78.45	94.99	**95.64**	94.26	42.18
100	84.64	92.45	88.36	86.35	78.49	94.99	**95.64**	94.26	43.09

MASPLAS: Rate of Fault Detection



Results using All Prioritization Metrics (log scale)

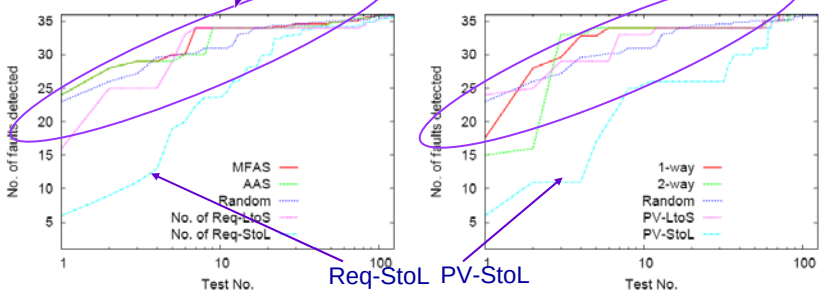
MASPLAS: Rate of Fault Detection as Measured by APFD

% of test suite run	Most Frequently Accessed request	All Accessed Sequence	No. of Requests Long to Short	No. of Requests Short to Long	Random	1-way	2-way	PVs Long to Short	PVs Short to Long
10	78.54	92.17	**95.12**	81.5	76.33	89.6	90.98	86.05	4.44
20	84.28	92.88	**95.12**	91.06	80.51	93.04	90.98	89.74	4.44
30	88.86	94.19	**95.12**	91.06	85.57	93.04	94.28	89.74	26.61
40	90.42	95.86	95.68	91.59	87.59	95.56	**97.06**	93.38	30.08
50	91.34	95.86	95.68	91.59	89.91	95.56	**97.06**	94.84	50.16
60	91.7	95.86	95.68	91.59	90.69	95.56	**97.06**	94.84	53.91
70	91.99	95.86	95.97	91.89	90.69	95.56	**97.06**	94.84	57
80	92.16	96.21	96.14	92.08	90.91	95.56	**97.06**	94.84	58.1
90	92.16	96.21	96.22	92.17	90.91	95.56	**97.06**	94.84	58.85
100	92.16	96.21	96.22	92.2	90.91	95.56	**97.06**	94.84	58.85

Book: Rate of Fault Detection

MFAS, AAS,
Random, Req-LtoS

1-way, 2-way,
PV-LtoS, Random



Frequency and Request-Length Based Techniques (log scale)

Parameter-value based Techniques (log scale)

Book: Rate of Fault Detection as Measured by APFD

% of test suite run	Most Frequently Accessed request	All Accessed Sequence	No. of Requests Long to short	No. of Requests Short to long	Random	1-way	2-way	PVs Long to Short	PVs Short to Long
10	93.33	93.13	92.96	70.04	90.34	**93.44**	93.22	93.11	70.13
20	93.79	93.13	92.96	86.09	93.7	**93.44**	93.22	93.11	70.13
30	**94.65**	93.56	92.96	88.15	94.52	93.44	93.22	93.11	78.17
40	**94.99**	94.26	92.96	88.91	94.86	93.44	93.22	93.11	79.86
50	**95.28**	95.16	92.96	88.91	94.86	94.96	94.69	93.11	84.12
60	**95.5**	95.41	92.96	89.15	95.11	96.13	94.69	94.47	86.73
70	95.9	95.41	93.74	89.54	95.27	**96.13**	95.62	95.56	86.73
80	**96.16**	95.69	94.11	89.81	95.56	**96.13**	95.62	95.56	86.73
90	**96.16**	95.69	94.18	89.92	95.56	**96.13**	95.62	95.56	86.73
100	**96.16**	95.69	94.27	89.94	95.57	**96.13**	95.62	95.56	86.73

All Applications: Execution Time

Execution Time: time taken to replay test suite
(Does not include time taken to detect faults, i.e., fault detection replay)

Application	1-way	2-way	PV-LtoS	PV-StoL
CPM	83.26 (813)	**38.88 (618)**	58.20 (746)	100 (889)
MASPLAS	40.24 (35)	**33.73 (36)**	42.01 (46)	97.04 (71)
Book	57.60 (907)	66.40 (1024)	60.80 (1002)	**54.40 (300)**

Percent of Test Suite Run (Execution time in seconds) for 100% Fault Detection

CPM: 2-way finds 100% of faults with 38% of test suite in 618 seconds
Masplas: 2-way finds 100% of faults with 33% of test suite in 36 seconds
Book: PV-StoL finds 100% of faults with 54% of test suite in 300 seconds

Summary of Results

- In two out of three applications studied
  - 2-way finds 100% of faults earliest with high APFD values
- Frequency-based metrics close second
- Length-LtoS, PV-LtoS comparable to other criteria
- Guidance to Testers
  - If large number of parameter-values present in application: select one of the interaction-based metrics
  - If testing certain functionality important: select frequency-based metrics
- Choosing right prioritization strategy can help find and fix faults quickly => time saving => cost saving

Conclusions and Future Work

- Presented length, frequency and parameter-value interaction-based prioritization techniques for web application test cases
- Empirically evaluated prioritization techniques for three subject web applications
- Interaction-based techniques and frequency-based techniques were effective at prioritizing web application test cases
- Future Work
  - Larger empirical study with more applications and other web application test cases that are in the form of HTTP requests
  - Additional frequency-based techniques
  - Hybrid prioritization techniques