Web Application Testing with Customized Test Requirements – An Experimental Comparison Study

Sreedevi Sampath[1], Sara Sprenkle[2],
Emily Gibson[2], Lori Pollock[2]

[1] UMBC
AN HONORS UNIVERSITY IN MARYLAND

[2] University of Delaware

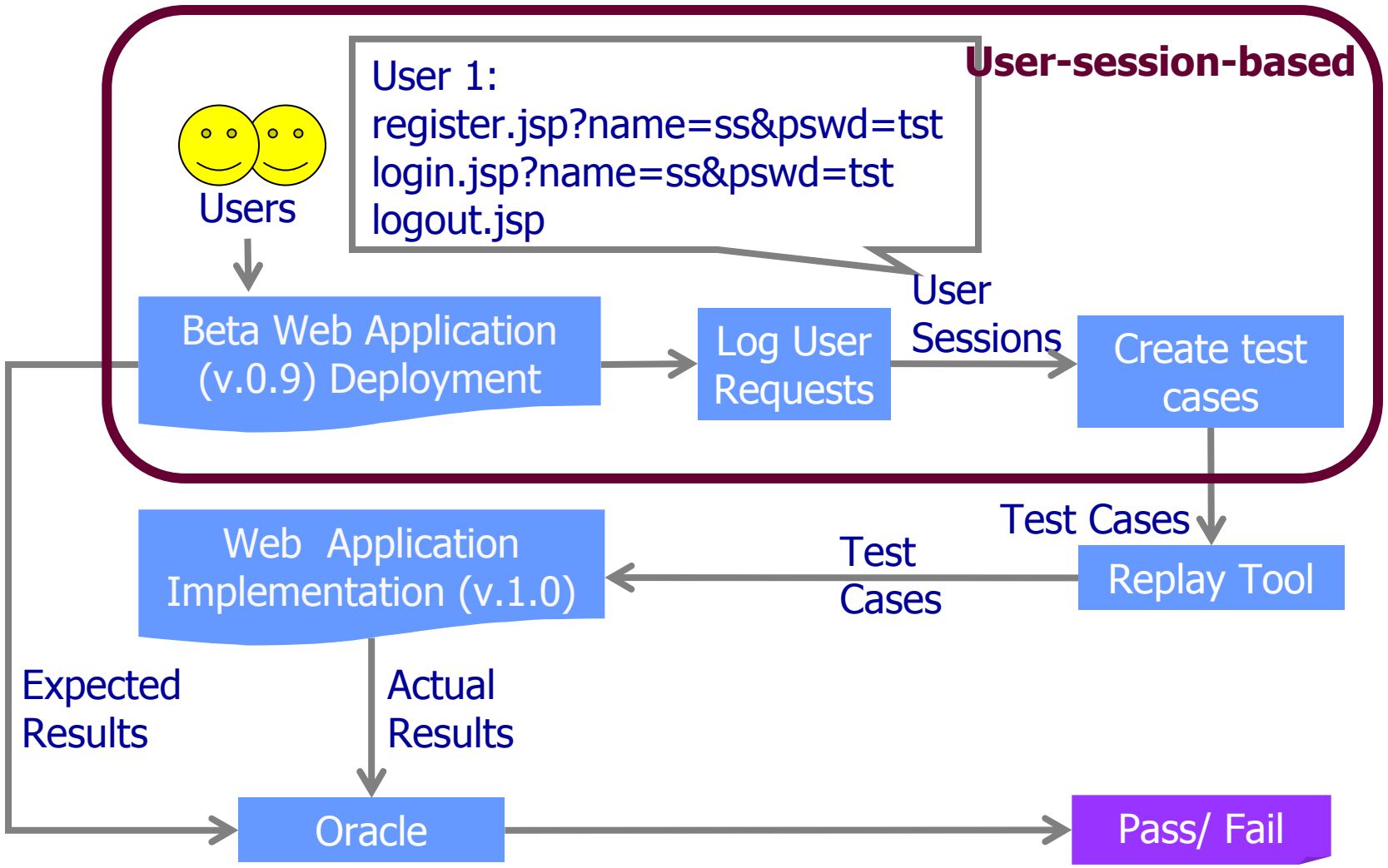
November 10, 2006

Need for Reliable Web Applications

- *Expedia* sells more than $35 million in tickets every week[1]
- In 2003, *amazon.co.uk* sold 64MB 200MHz HP iPAQ's valued at £ 275 for £ 7.32[2]
- Huge losses on web site failure[3]
  - Financial services: $6.5 million per hour
  - Credit card sales applications: $2.4 million per hour
  - Media companies: $150,000 per hour
- Large number of failures during maintenance[4]

1. e-Business 2.0: Roadmap for Success (2nd Edition) by M. Robinson, D. Tapscott, R. Kalakota . 2000
2. The 10 worst web application failures in Information Age by Kenny MacIver. May 2003
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

User-session-based Testing Process

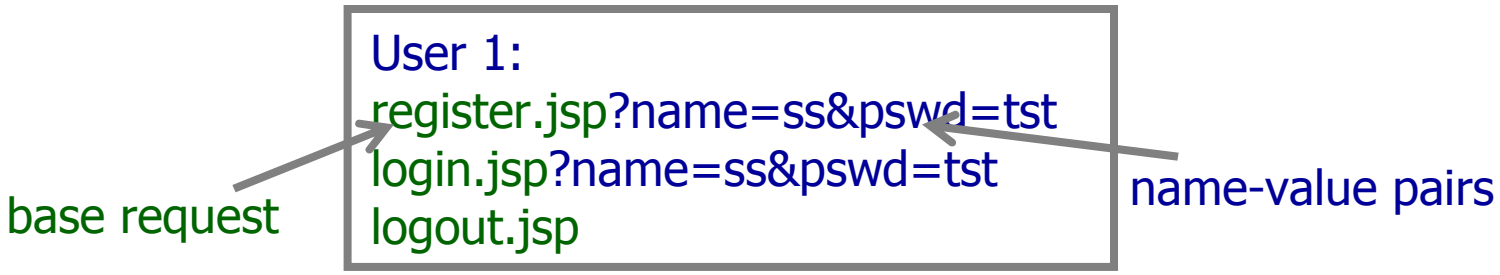


Measure Quality of Test Suites

- Test requirement coverage
  - When to stop testing
  - Which test cases to select
  - How to reduce a test suite
- Control and data flow-based requirements
  - Statement, Method, Branch, Def-use
  - Covering all the statement requirements ensures the statement coverage criterion is satisfied

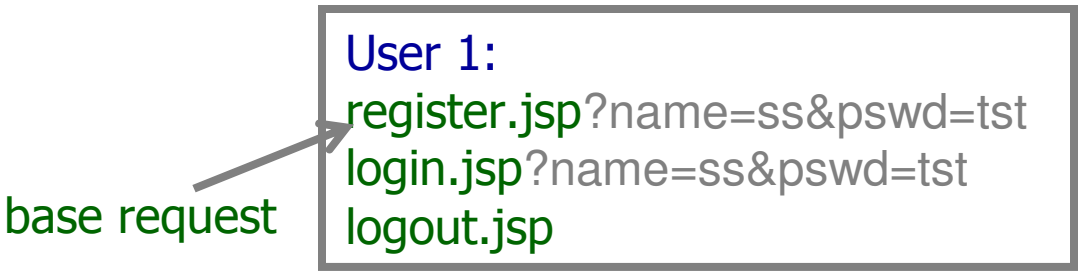
Measure Quality of Test Suites

- Test requirement coverage
  - When to stop testing
  - Which test cases to select
  - How to reduce a test suite
- Coverage and data flow-based requirements
- We proposed usage-based test requirements
  - Derived from usage-data



Measure Quality of Test Suites

- Test requirement coverage
  - When to stop testing
  - Which test cases to select
  - How to reduce a test suite
- Coverage and data flow-based requirements
- We proposed usage-based test requirements
  - Derived from usage-data
  - Requirement base



Measure Quality of Test Suites

- Test requirement coverage
  - When to stop testing
  - Which test cases to select
  - How to reduce a test suite
- Coverage and data flow-based requirements
- We proposed usage-based test requirements
  - Derived from usage-data
  - Requirement base
  - A reduced suite that provides full base coverage indicates that for every base request b covered by the original suite, there is at least one test case in the reduced suite that covers b.

Customized Test Requirement: base

- base: create reduced suite that covers all base requests

- Example test case

< GET login.jsp?name=xxx&pswd=yyy,
GET shop.jsp?item_no=aaa&book_name=ccc&price=60 >

- Requirements to be satisfied for base

{GET login.jsp, GET shop.jsp}

Test Suite Reduction With Requirement base

- Original Test Cases

User 1:
GET login.jsp?name=xxx&pswd=yyy
GET shop.jsp?item_no=1&book_name=DaVinciCode&price=60

User 2:
GET login.jsp?name=pp&pswd=incorrectpassword
GET shop.jsp?item_no=2&book_name=Elizabeth&price=160

- Requirements to be satisfied for base

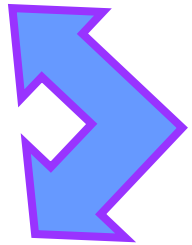
{GET login.jsp, GET shop.jsp}

Test Suite Reduction With Requirement base

- Original Test Cases

User 1:
GET login.jsp
GET shop.jsp

User 2:
GET login.jsp
GET shop.jsp

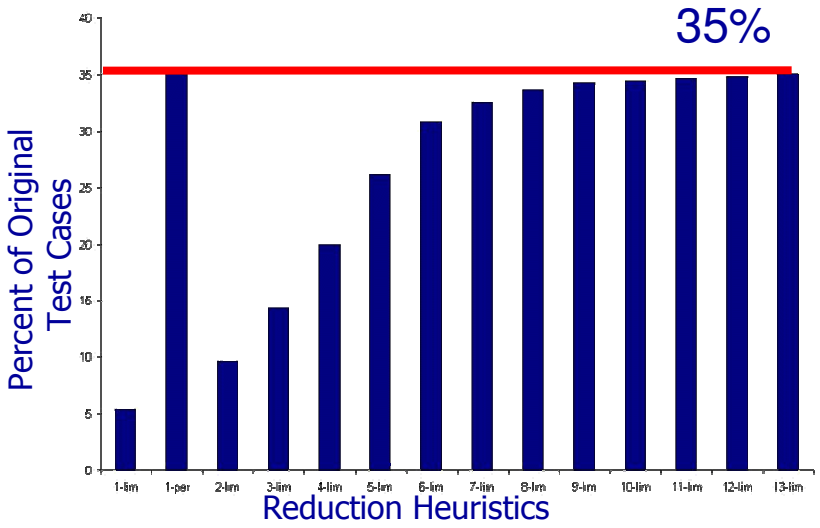


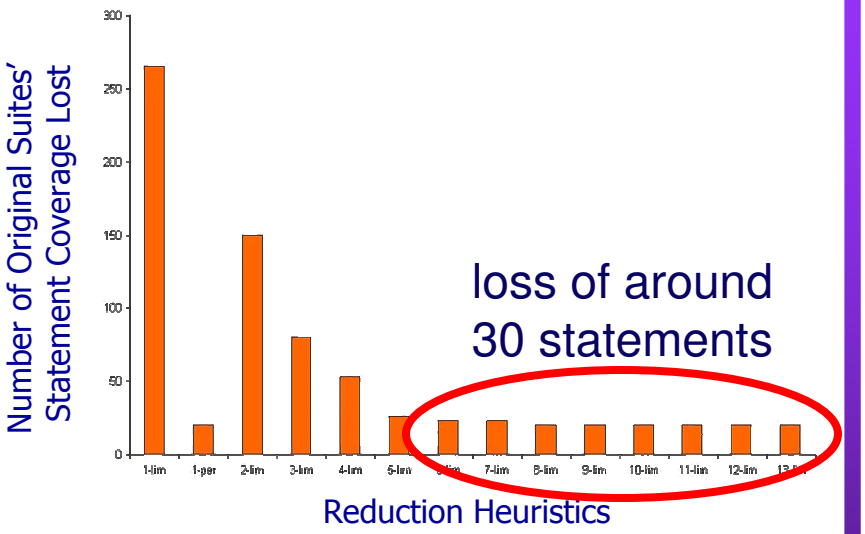
Viewed as the same test case
Only one test case is selected

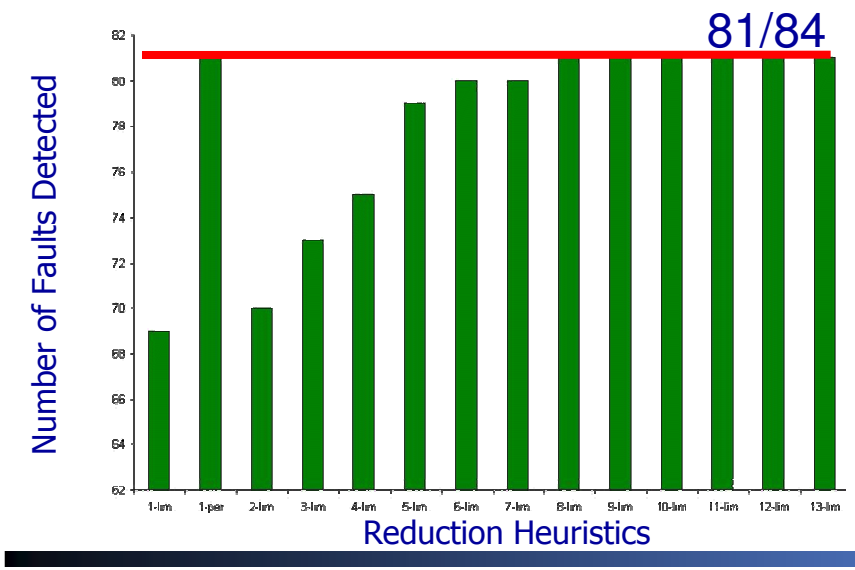
- Requirements to be satisfied for base

{GET login.jsp, GET shop.jsp}

Results From Previous Work







Total faults seeded:135

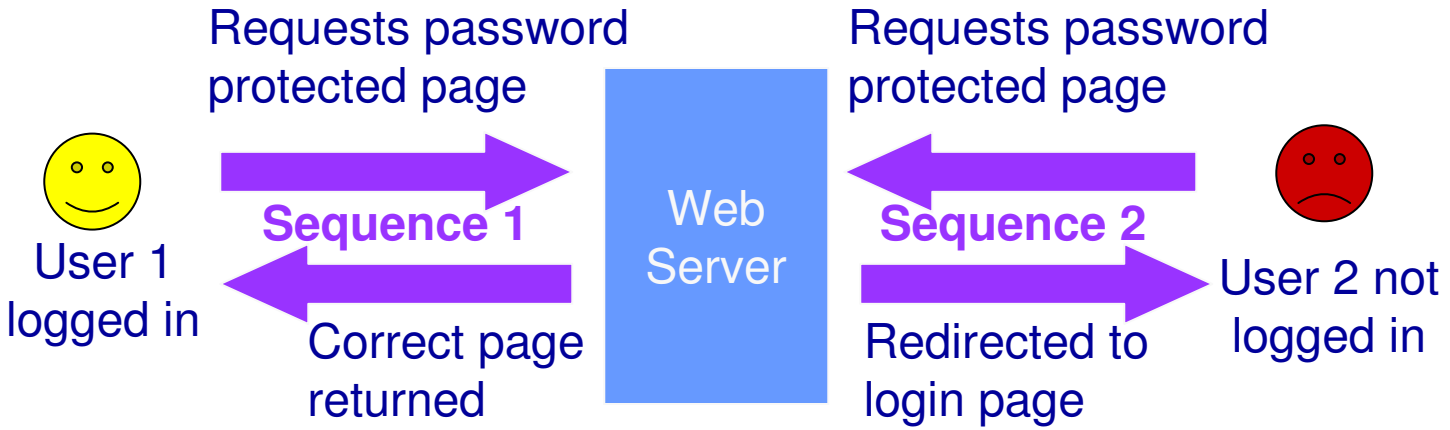
Summary:

- base creates reduced suites with large percent reduction
- But the reduced suites are not as effective as the original suite

Motivate need for alternate requirements

On further analysis we found that

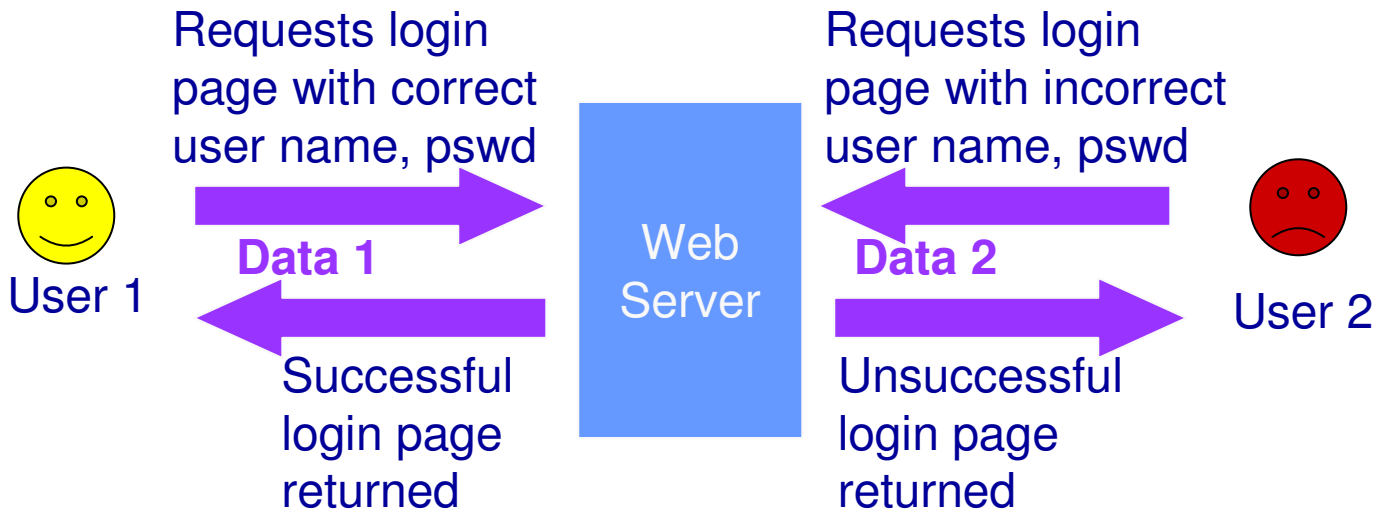
- Certain faults are detected by a particular request sequence



- Unless sequence 2 is in the reduced suite, redirection code executed by User 2 will not be covered by reduced suite

On further analysis we found that

- Data associated with the request affects test suite effectiveness



- Unless Data 2 is present in the reduced suite, code executed by User 2 on unsuccessful login will not be covered by reduced suite

Customized Test Requirements: seqk

- seqk: cover all size *k* sequences of base requests

- Example test case

< GET login.jsp?name=xxx&pswd=yyy,
GETshop.jsp?item_no=aaa&book_name=ccc&price=60 >

- Requirements to be satisfied for seq2

for k=2, the sequence
{<GET login.jsp, GET shop.jsp>}

Customized Test Requirements: name

- name: cover all base requests and names

- Example test case

< GET login.jsp?name=xxx&pswd=yyy,
GETshop.jsp?item_no=aaa&book_name=ccc&price=60 >

- Requirements to be satisfied for name

{GET login.jsp?name&pswd,
GET shop.jsp?item_no&book_name&price}

Customized Test Requirements: namevalue

- namevalue: cover all base requests and name and value pairs

- Example test case

< GET login.jsp?name=xxx&pswd=yyy,
GETshop.jsp?item_no=aaa&book_name=ccc&price=60 >

- Requirements to be satisfied for namevalue

{GET login.jsp?name=xxx&pswd=yyy,
GET shop.jsp?item_no=aaa&book_name=ccc&price=60}

Customized Test Requirements: seqkname

- seqkname: cover all size k sequences of base requests and names

- Example test case

< GET login.jsp?name=xxx&pswd=yyy,
GETshop.jsp?item_no=aaa&book_name=ccc&price=60 >

- Requirements to be satisfied for seq2name

for k = 2, the sequence
{<GET login.jsp?name&pswd,
GET shop.jsp?item_no&book_name&price>}

Experimental Study

- Goal: Evaluate tradeoffs between requirements
- Metrics
  - reduced test suite size
  - program coverage effectiveness
  - fault detection effectiveness
  - test suite replay costs
  - test suite reduction time and space costs

Expected Tradeoffs

- More data/context associated with requirement

   More complex the requirement

- Expect suite size, program coverage, fault detection to follow the trend

  base <= seq2 <= seq2name

  base <= name <= namevalue

  name <= seq2name

Subject Applications

Metrics	Masplas	Bookstore	CPM	DSpace
Technology	Java Servlets, JSP, MySql database	JSP, MySql database	Java Servlets, Files data store	Java Servlets, JSP, PostGreSQL and files data store
Non-comment Lines of Code	999	7791	9300	61729
Classes	9	11	75	508
Methods	42	385	172	4233
Faults	30	39	135	45
Test Cases (User sessions)	169	125	890	1342

Test Suite Characteristics

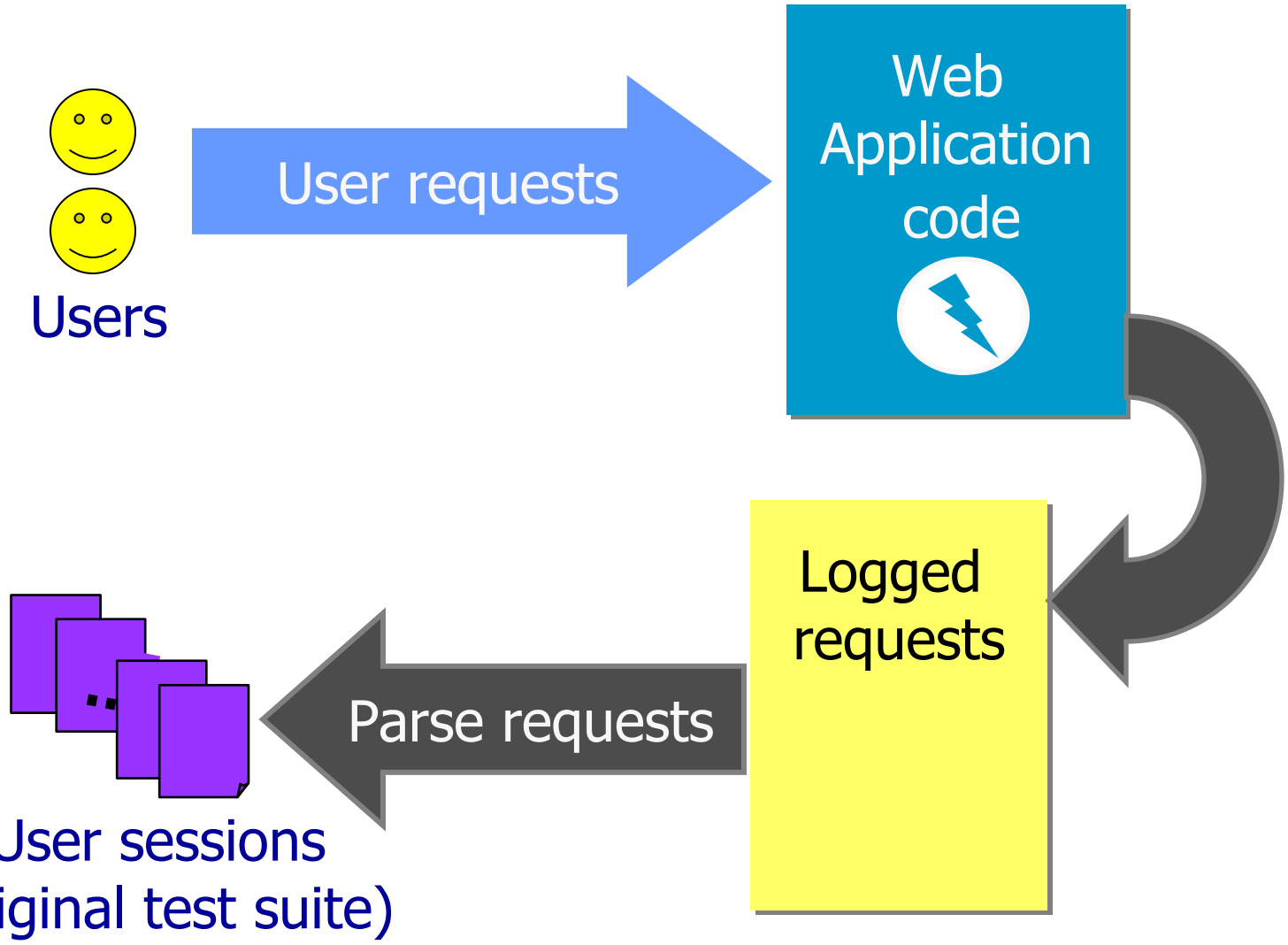
Application	# of test cases
Masplas	169
Book	125
CPM_ALL	890
CPM_Y04	261
CPM_Y05	629
CPM_A04	58
CPM_F05	203
CPM_M05	105

Application	# of test cases
CPM_A05	168
CPM_D05	356
DSpace	1342

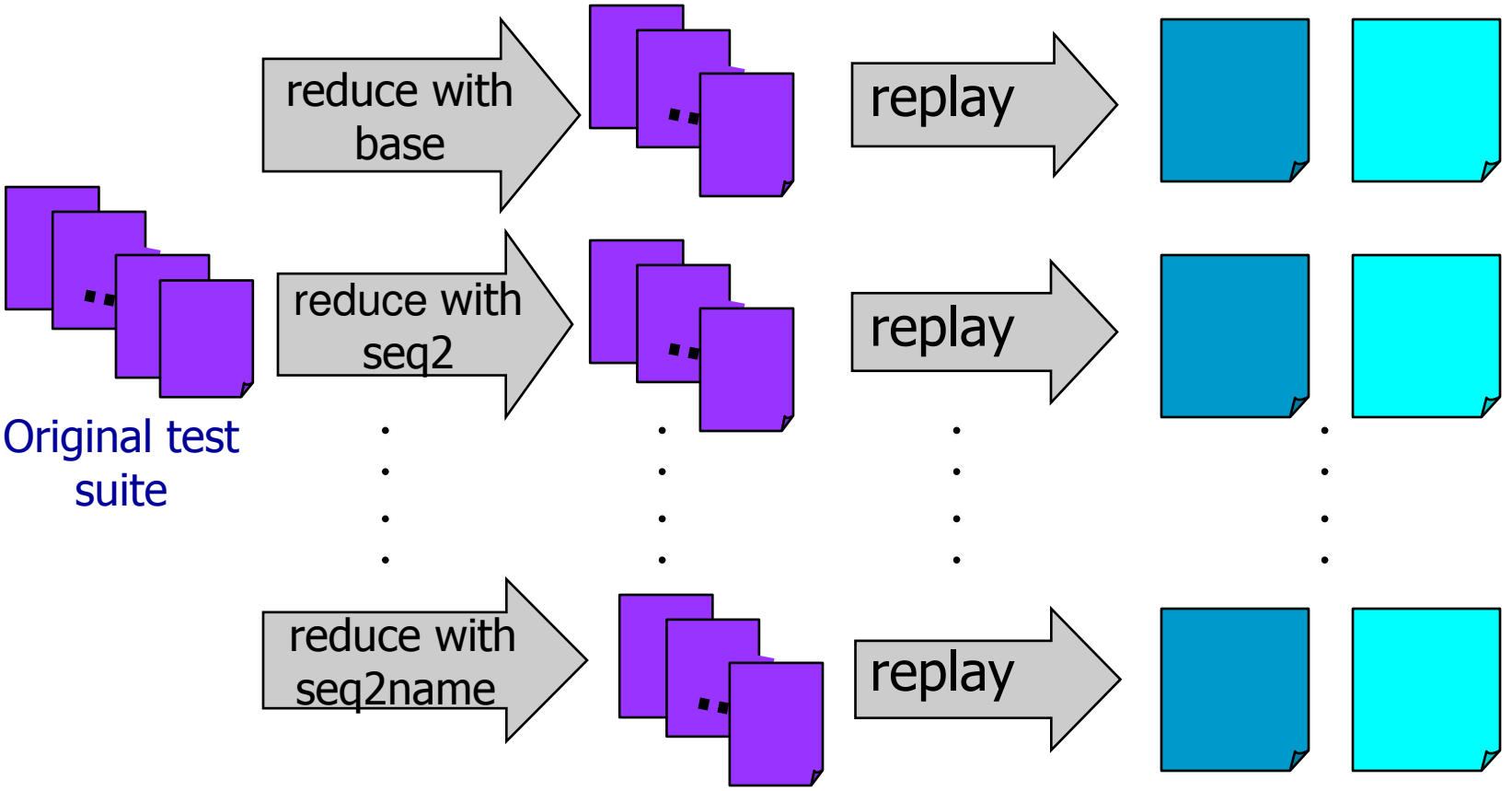
Reduction Techniques

- Our concept analysis-based test suite reduction algorithm [ASE 2004]
  - criterion: cover all requirements covered by original suite, while still covering the set of requirements covered by the original suite
- Harrold, Gupta and Soffa's test suite reduction algorithm [TOSEM 1993]
  - criterion: Cover all requirements covered by the reduced suite

Case Study: Methodology (1)



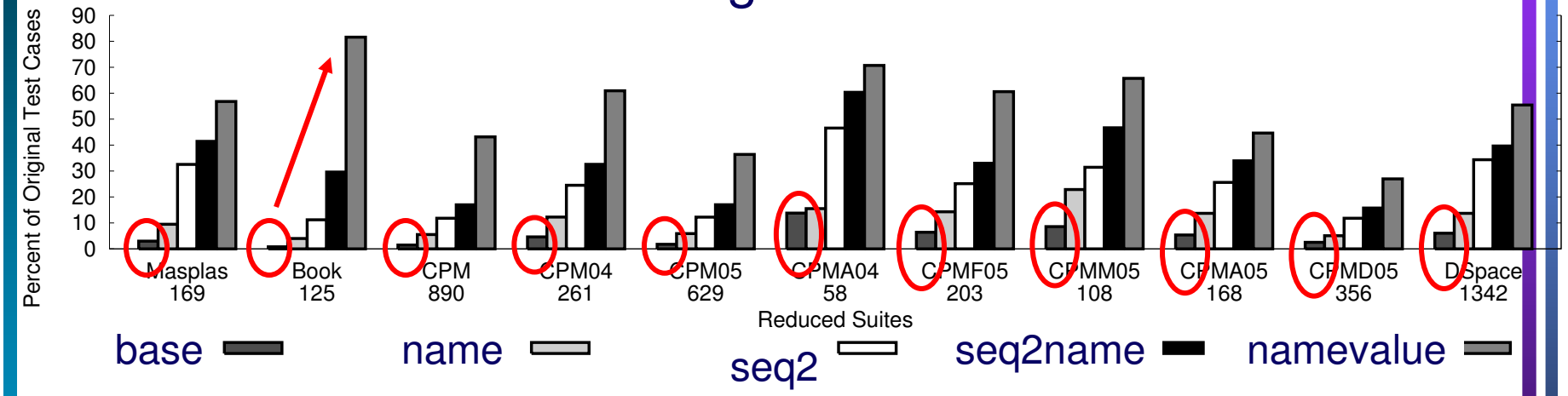
Case Study: Methodology (2)



Results: Reduced Test Suite Size

X-axis: Reduced test suites using the different requirements

Y-axis: $\frac{\text{\# of test cases in reduced test suite}}{\text{\# of test cases in original test suite}} * 100$



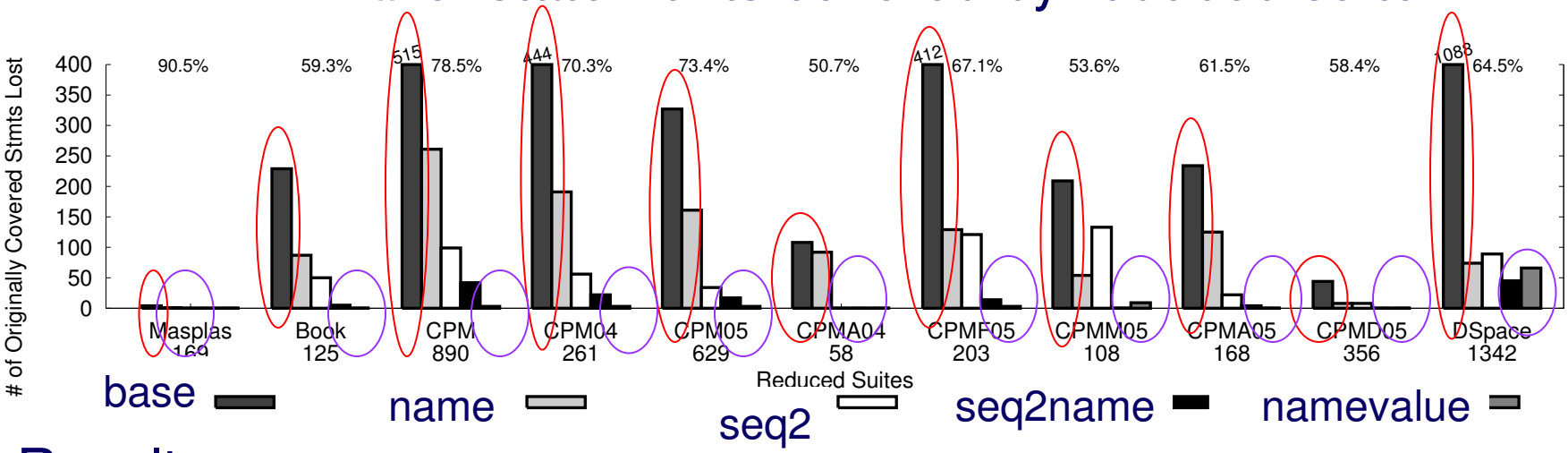
Results:

- base selects the smallest reduced suite
- With increase in requirement complexity, the reduced test suite size increases

Results: Program Coverage Effectiveness

X-axis: Reduced test suites using the different requirements

Y-axis: # of statements covered by original suite –
# of statements covered by reduced suite



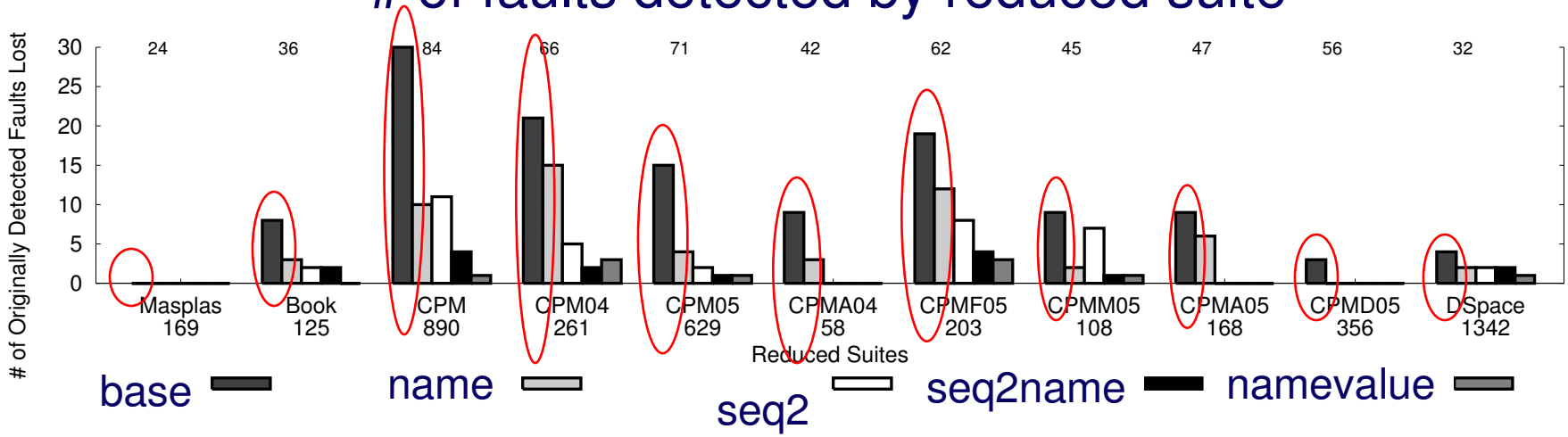
Results:

- base loses the most program code
- Requirement has an impact on corresponding code covered
- With increase in requirement complexity,
  the statement loss decreases

Results: Fault Detection Effectiveness

X-axis: Reduced test suites using the different requirements

Y-axis: # of faults detected by original suite –
# of faults detected by reduced suite



Results:

- base has the highest loss in fault detection
- With increase in requirement complexity,
  the number of faults lost decreases

Summary of Statistical Analysis

- To measure cost-effectiveness of the requirements we defined a figure of merit, fom = redux * cvg * fd
  - redux: percent reduction
  - cvg: percent coverage
  - fd: percent fault detection
- Higher the fom, more cost-effective the reduced suites created by requirement
- Requirements base and name emerged winners across all four subject applications

Recommendations

- For applications that are control dependent on input names and values
  - use requirements name, namevalue, seq2name
- For applications that have specific user patterns intended by developer
  - use requirements seq2, seq2name

Related Work

- Structural [Andrews et al., Lucca et al., Liu et al., Ricca et al.], functional [link/form testers] and user-session-based web testing [Elbaum et al., Sampath et al.] approaches exist
- Approach to generate test cases that bypass client side validation [Offutt et al.]
  - Our approach generates test cases that represent common application usage
  - Offutt et al. generate test cases that represent less normal, malicious usage

Conclusions

- Defined a set of test requirements and experimentally evaluated their applicability to test case selection
- Results indicate that more the data/context associated with the requirement, higher the effectiveness of the reduced suite
- From statistical analysis: requirements name and base consistently find small, effective reduced suites

Future Work

- Classify application usage and evaluate applicability of requirements for test case selection
- Evaluate test requirements with more web applications