Integrating Customized Test Requirements with Traditional Requirements in Web Application Testing

Sreedevi Sampath, Sara Sprenkle
Emily Gibson and Lori Pollock
University of Delaware
July 17, 2006

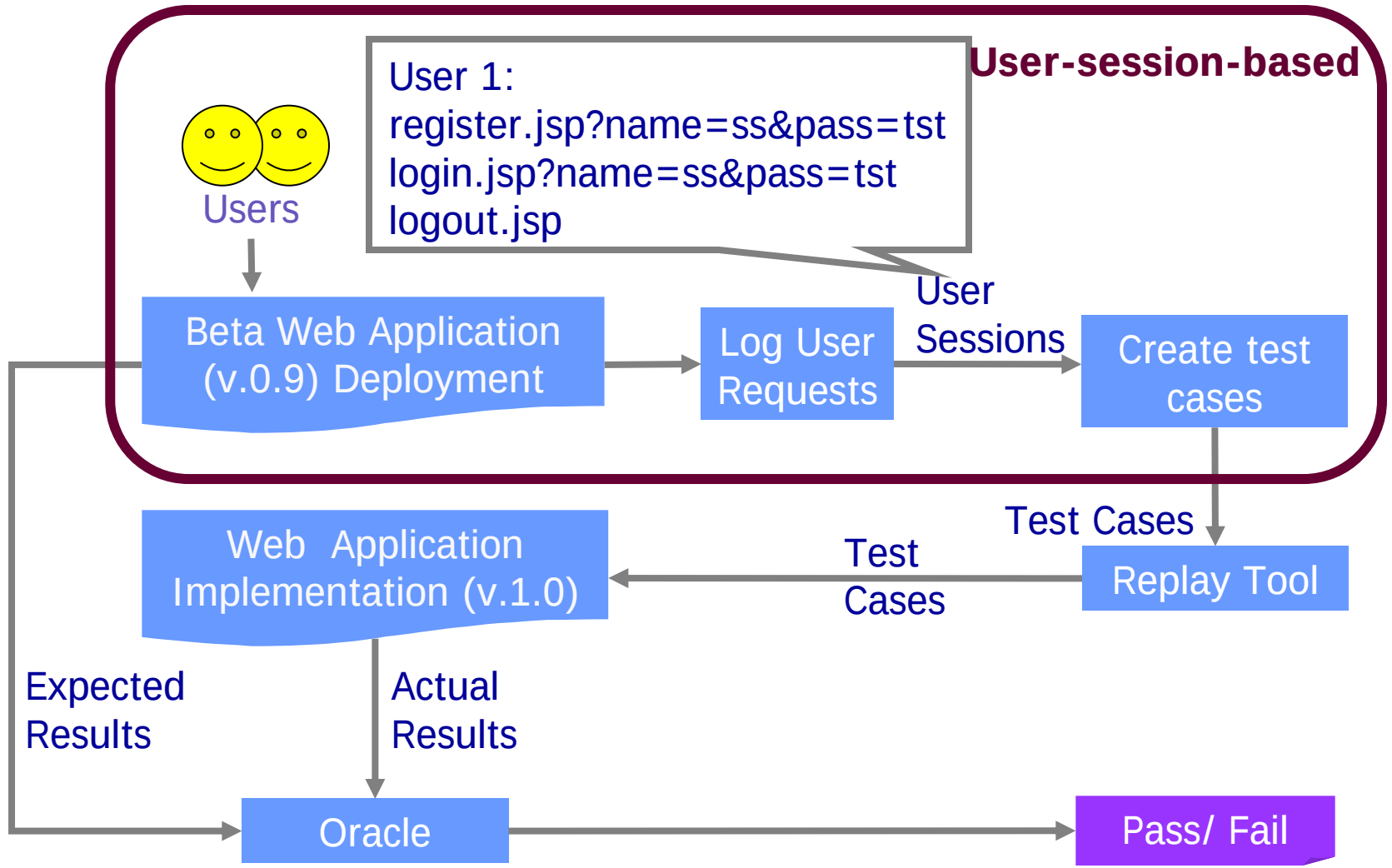
Workshop on Testing, Analysis and Verification of Web Applications and Services (TAV-WEB)

Need for Reliable Web Applications

- *Expedia* sells more than $35 million in tickets every week[1]
- In 1999 *KBKids.com* $10 off 30 dollars or cents?[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. More Web sites turn to test tools by Carol Sliwa in CNN.com. 1999
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



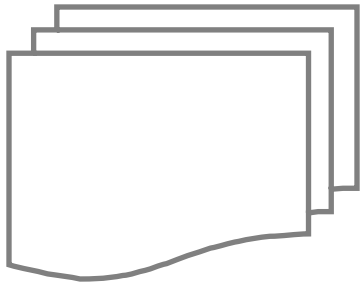
Measure Quality of Test Suites

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

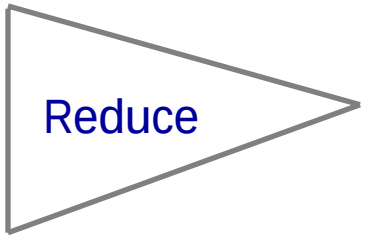
Program Coverage-based Test Requirement

Requirement: statement

Create reduced suite that covers all statements in code covered by original suite



Original Test Suite
(set of test cases)



Reduced Test Suite

Advantage

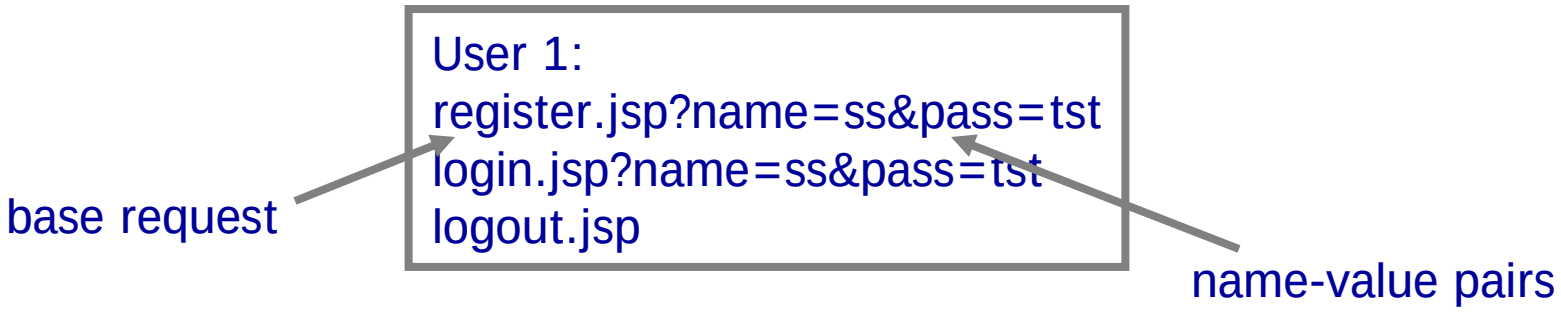
\+ Guaranteed program coverage by reduced suite

Disadvantage

\- Expensive to execute the original test suite prior to reduction

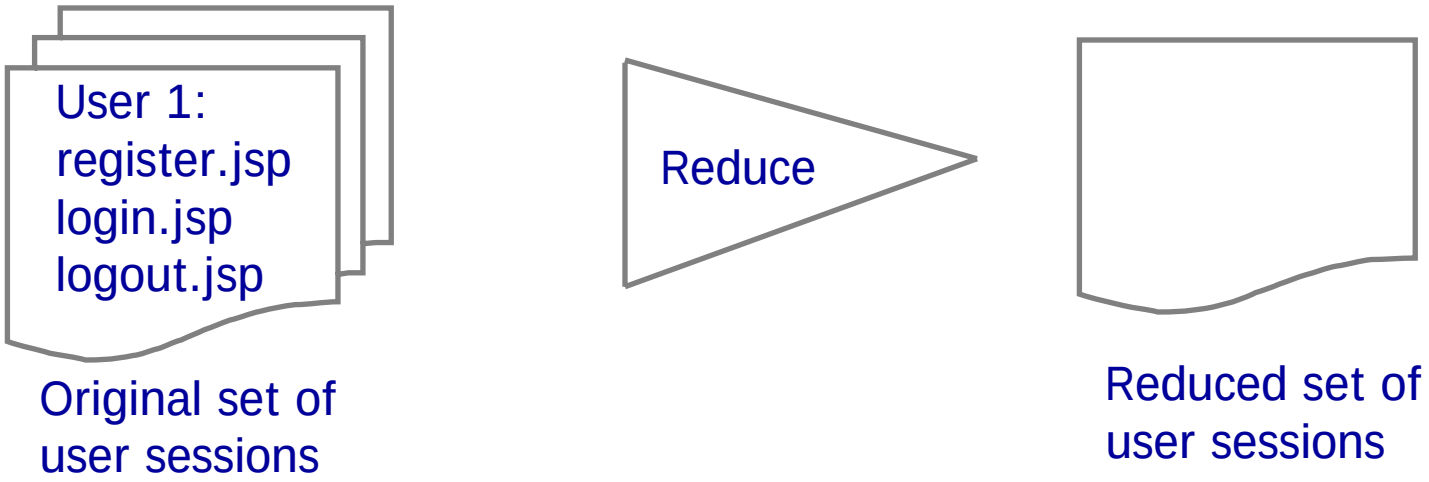
Measure Quality of Test Suites

- 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
  - base, name, seq2, namevalue, seq2name



Usage-based Test Requirement

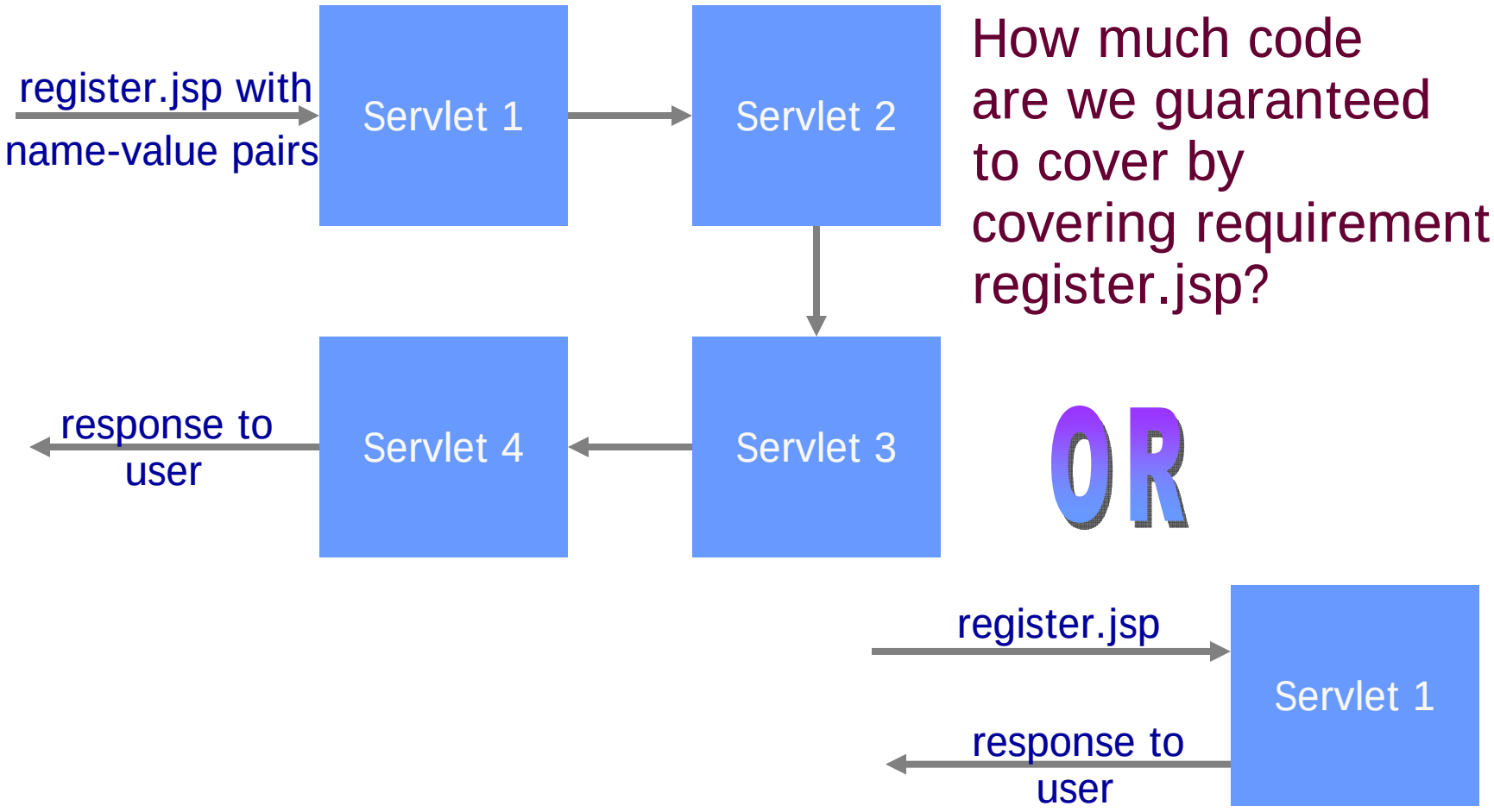
Requirement: base request
Create reduced suite that covers all base requests covered by original suite



Advantages

+ Information present in the user session itself
+ No need to execute original suite, cost-effective

Disadvantage of Usage-based Test Requirements



Integrate Program-based and Usage-based Requirements

We want to achieve a balance between

Cost of generating reduced suite and size of reduced suite

Program coverage and fault detection effectiveness of reduced suite

Our Research: Strategies for Integration

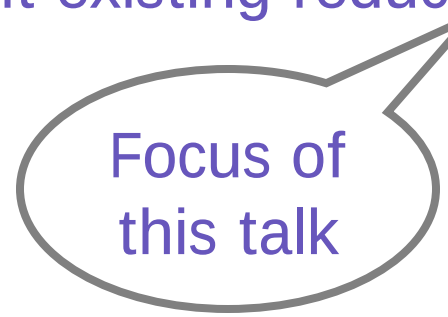
Integrate program coverage-based and usage-based test requirements to

1. Compare test suites to identify the better suite
2. Combine test requirements for reduction
3. Augment existing reduction algorithm

Our Research: Strategies for Integration

Integrate program coverage-based and usage-based test requirements to

1. Compare test suites to identify the better suite
2. Combine test requirements for reduction
3. Augment existing reduction algorithm



Usage-based Test Requirement: seqk

- Example test case

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

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

  for k=2,

  <{GET login.jsp, GET shop.jsp}>

Usage-based Test Requirement: name

- Example test case

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

- name: cover all base requests and names

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

Usage-based Test Requirement: namevalue

- Example test case

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

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

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

Usage-based Test Requirement: seqkname

- Example test case

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

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

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

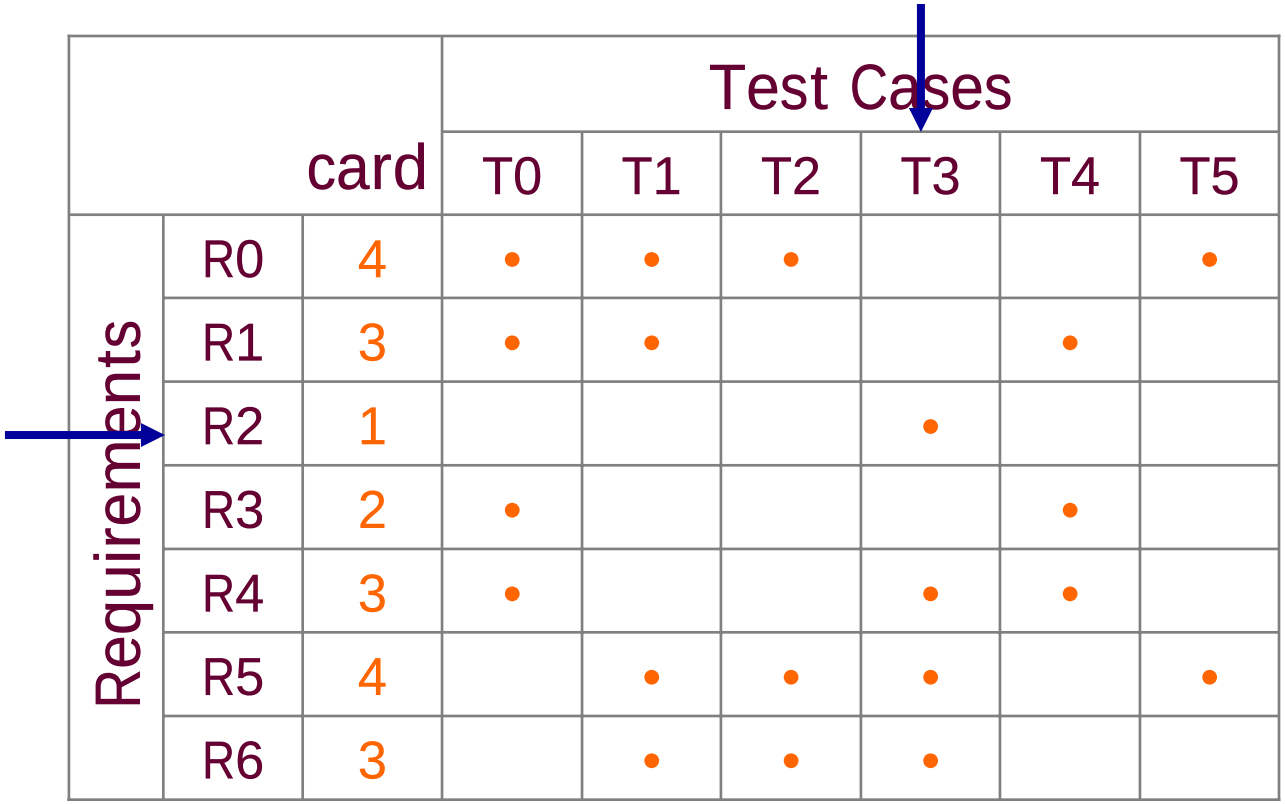
Traditional HGS

- Augment HGS [Harrold et al.] with usage-based requirements as tie breakers
- Select next test case to cover the least-covered requirement

Cardinality: number of test cases that cover each requirement

		card	Test Cases					
			T0	T1	T2	T3	T4	T5
Requirements	R0	4	⊙	⊙	⊙			⊙
	R1	3	•	•			•	
	R2	1				•		
	R3	2	•				•	
	R4	3	•			•	•	
	R5	4		•	•	•		•
	R6	3		•	•	•		

Traditional HGS



Start with requirement with least cardinality

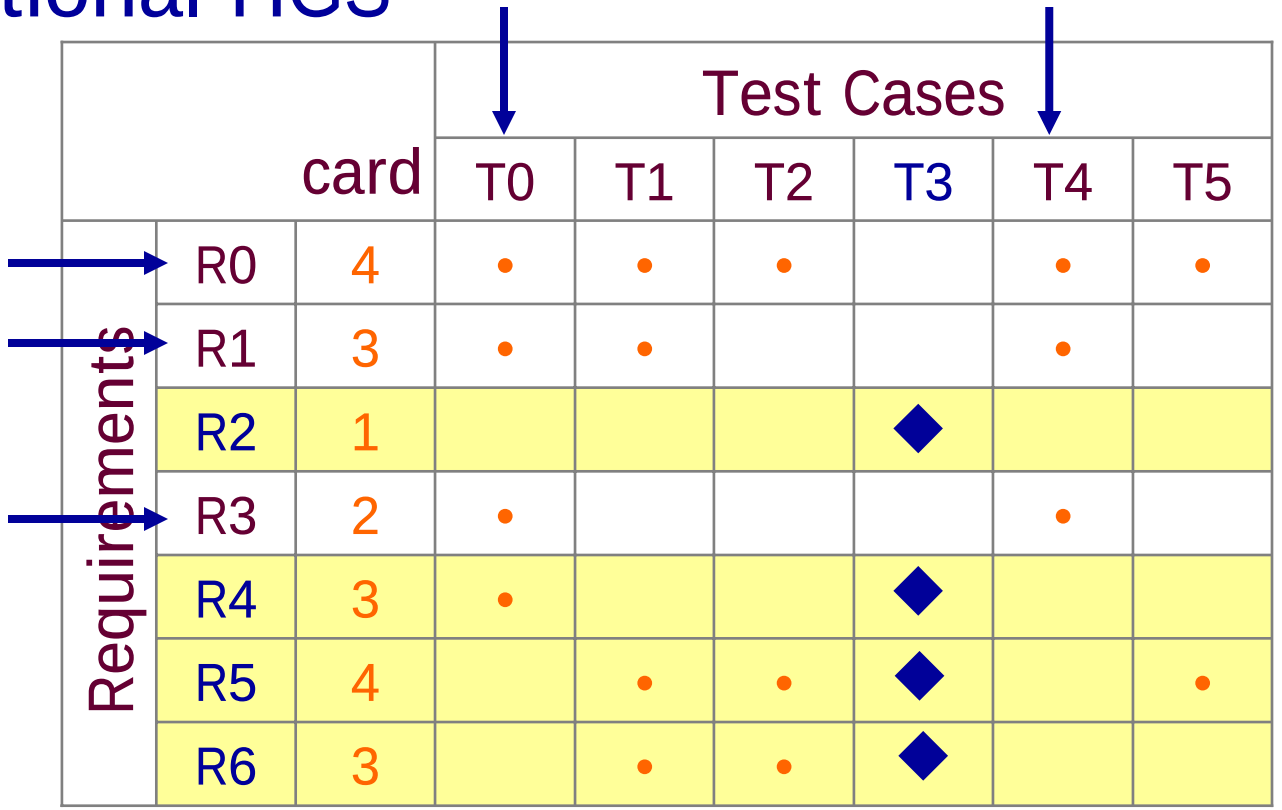
Only T3 covers R2

Traditional HGS

		card	Test Cases					
			T0	T1	T2	T3	T4	T5
Requirements	R0	4	•	•	•			•
	R1	3	•	•			•	
	R2	1				◆		
	R3	2	•				•	
	R4	3	•			◆	•	
	R5	4		•	•	◆		•
	R6	3		•	•	◆		

Choosing T3 covers R2, R4, R5, R6

Traditional HGS



- R3 is covered by two test cases, T0 and T4
- Choose test case that covers most uncovered requirements
  - Tie between T0 and T4, Go to next higher cardinality
  - Eventually, select randomly from tied test cases

Modified HGS

		card	Test Cases T0	T1	T2	T3	T4	T5
Requirements	R0	4	•	•	•		•	•
	R1	3	•	•			•	
	R2	1				◆		
	R3	2	•				•	
	R4	3	•			◆		
	R5	4		•	•	◆		•
	R6	3		•	•	◆		

Usage-based requirement coverage

T0

GET login.jsp
GET shop.jsp

T4

GET login.jsp

- Select T0 for the reduced suite
- More effective when large number of ties encountered

Case Study

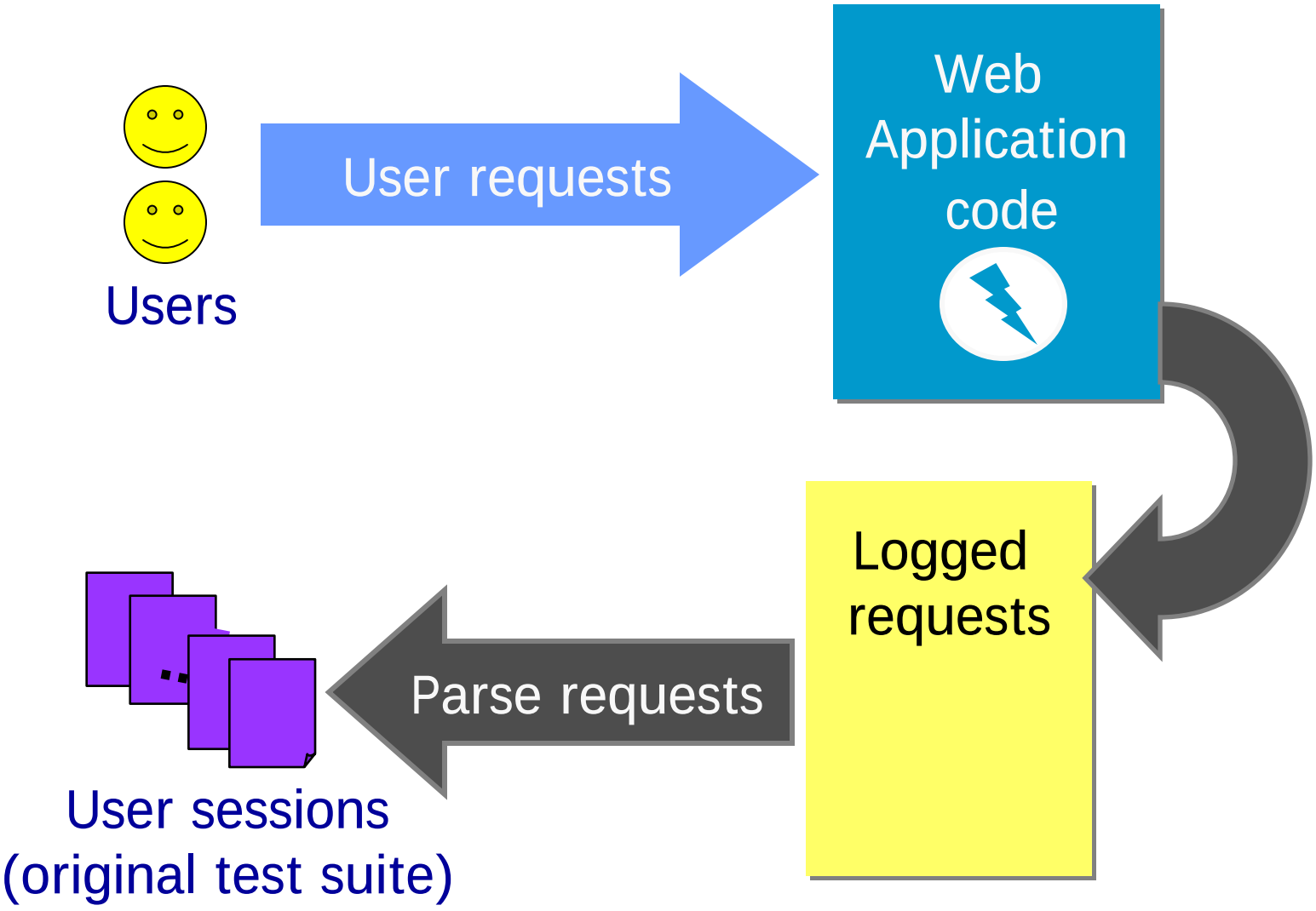
- Compare reduced suites from
  - Traditional HGS with program coverage-based requirements
  - Traditional HGS with usage-based requirements
  - Modified HGS with program coverage-based requirement (method) and usage-based tie breakers
- Metrics
  - Reduced suite size
  - Program coverage
  - Fault detection

Subject Application

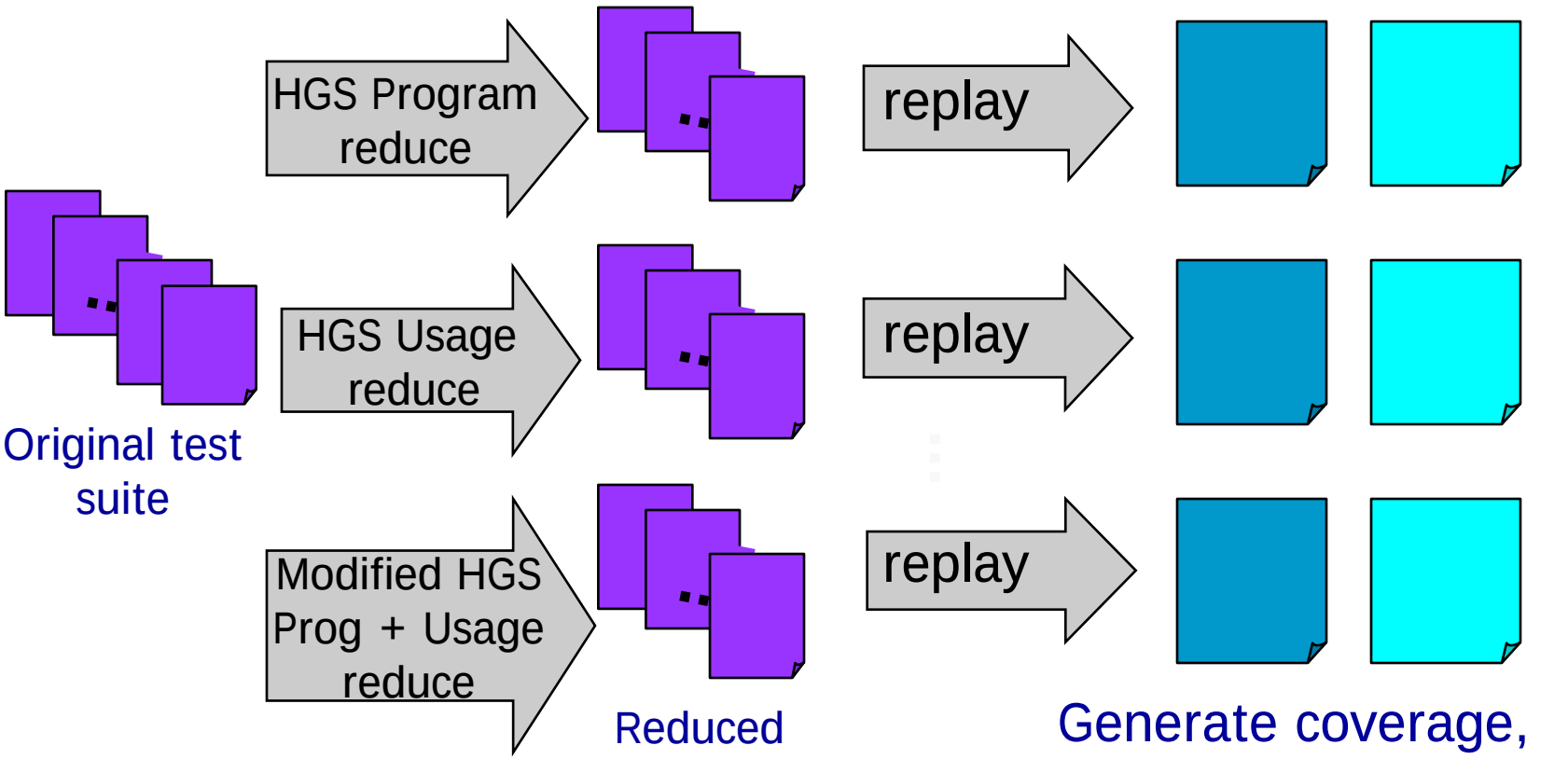
Course Project Manager (CPM)

Number of classes	75
Number of methods	172
Non-commented LOC	8947
Number of seeded faults	135
Number of user sessions	890
Total number of URLs	12,352

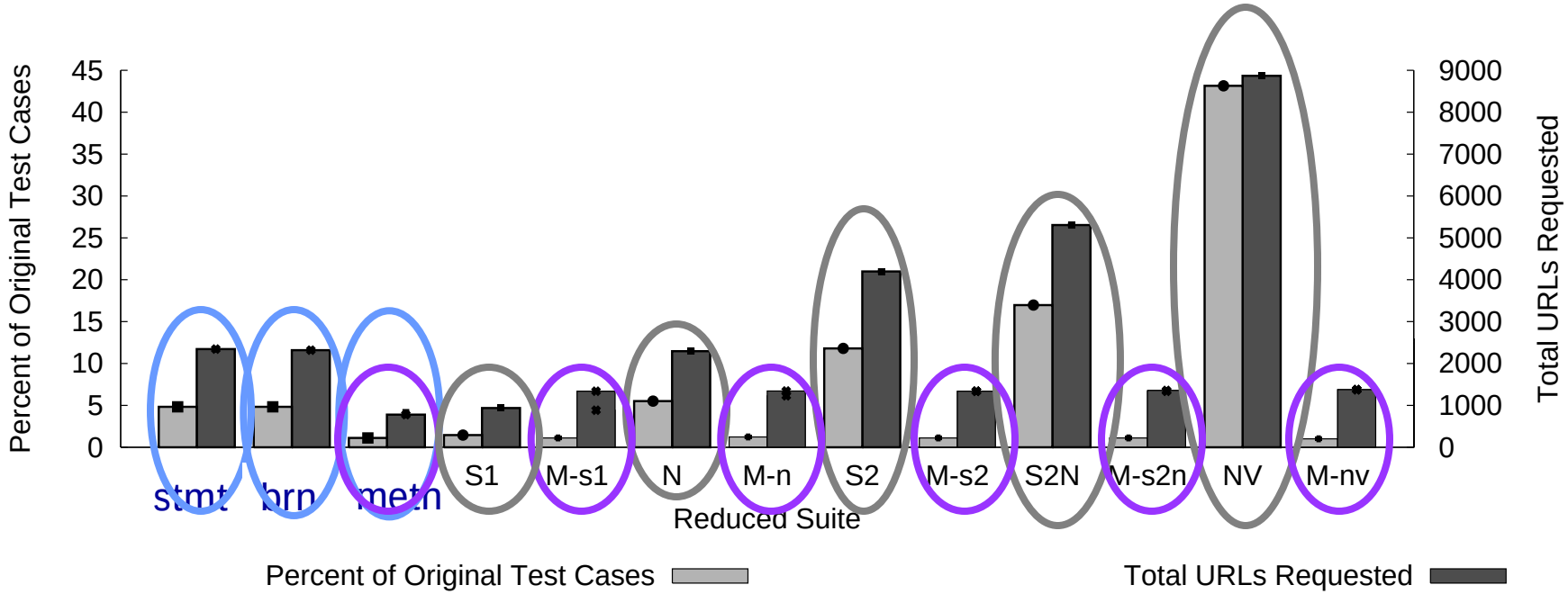
Case Study: Methodology (1)



Case Study: Methodology (2)



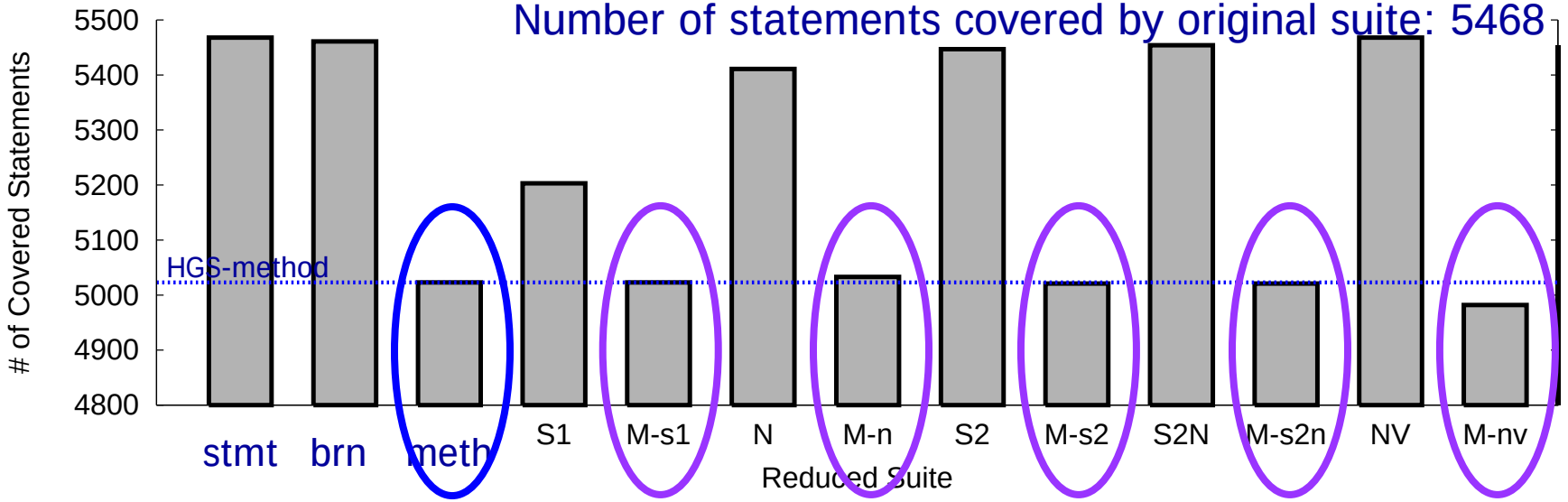
Results: Reduced Test Suite Size



Modified HGS		
Label in graph	**Prog cvg req.**	**Tie breaker**
M-s1	method	base request
M-n	method	name
M-s2	method	seq2
M-s2n	method	seq2name
M-nv	method	namevalue

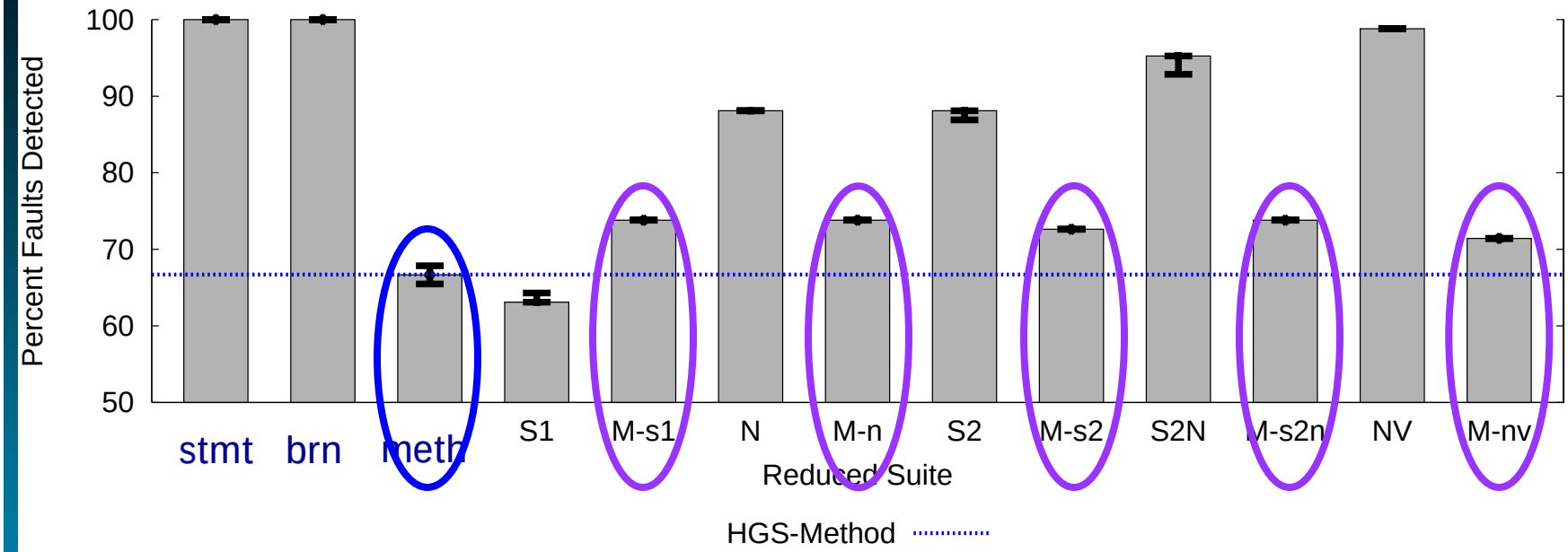
Results: Program Coverage Effectiveness

Total Number of statements in CPM: 6966
Number of statements covered by original suite: 5468



Modified HGS		
Label in graph	**Prog cvg req.**	**Tie breaker**
M-s1	method	base request
M-n	method	name
M-s2	method	seq2
M-s2n	method	seq2name
M-nv	method	namevalue

Results: Fault Detection Effectiveness



Modified HGS		
Label in graph	**Prog cvg req.**	**Tie breaker**
M-s1	method	base request
M-n	method	name
M-s2	method	seq2
M-s2n	method	seq2name
M-nv	method	namevalue

Results: Summary

- Fault detection of Modified HGS better than Traditional HGS for same suite size and generation cost
- Usage-based requirement alone more effective than using Modified HGS with program-based and usage-based tie breakers but test suite size increases
- Type of usage-based requirement -- no effect on effectiveness of Modified HGS reduced suites

Conclusions and Future Work

- Presented three strategies to integrate usage-based and program-based requirements in paper
- Experimentally evaluated the strategies
  - Instead of using method requirement, use customized requirement alone
  - Combining requirements better than HGS-method
  - Modified HGS better than traditional HGS
- Future Work
  - Extend study to other applications
  - Augment test cases from models of application with usage-based requirements