Automated Oracle Comparators for Testing Web Applications

Sara Sprenkle

Washington and Lee University

Lori Pollock

University of Delaware

Holly Esquivel

The University of Wisconsin Madison

Barbara Hazelwood

Cisco

Stacey Ecott

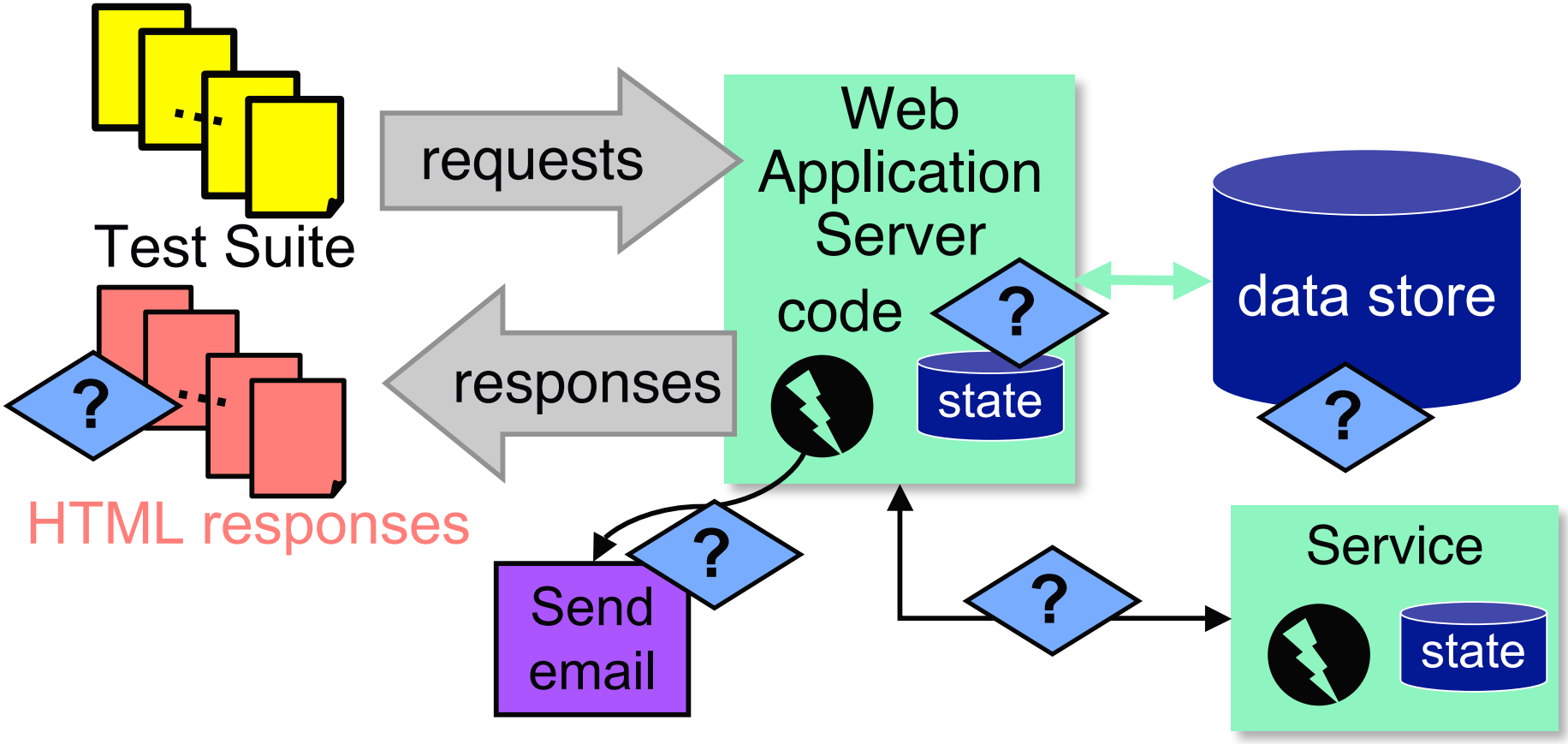
athenahealth

Testing Web Applications

- 772 million people online in May 2007*
- Amazon.com had 142K *unique* visitors, on average visit 3.1 times, in May 2007*
  - Earned $10.7 billion in revenue in 2006
- Need reliable web applications
  - Errors are seen by many, quickly
  - Frequent maintenance/update cycles
  - Need fast, effective oracles

* Source: comScore

Oracles for Web Applications

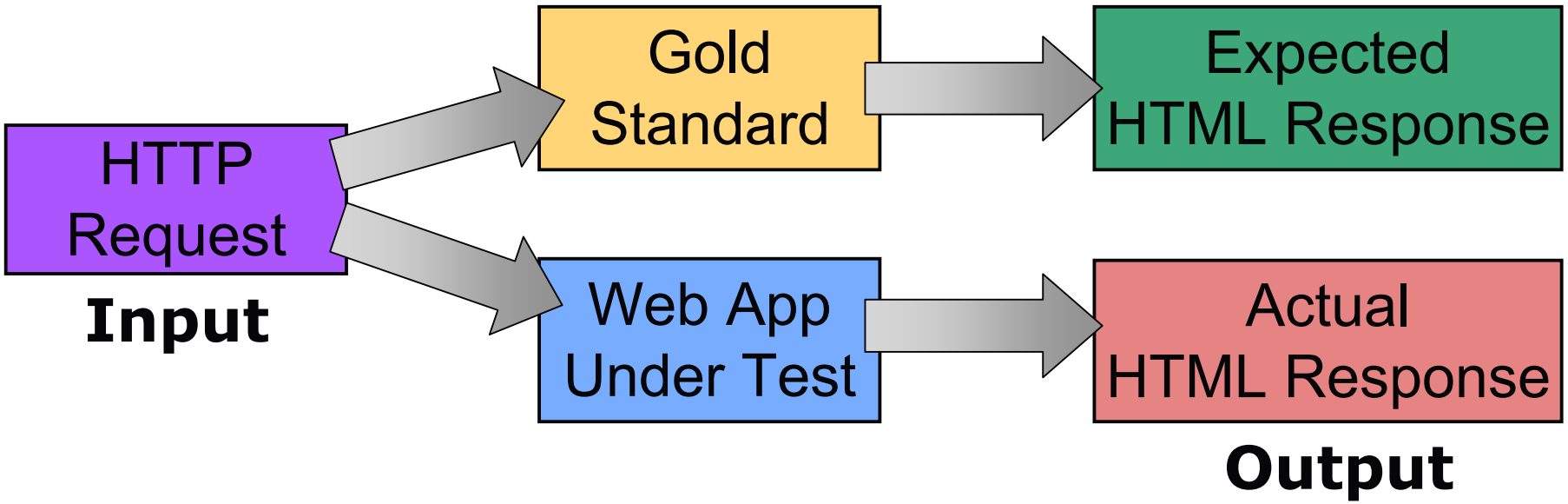


- Challenge: Low observability of outputs
- Solution: Use “pseudo-oracle” [Weyuker 82] on HTML
  - HTML: Common output for web applications

Overview of Contributions

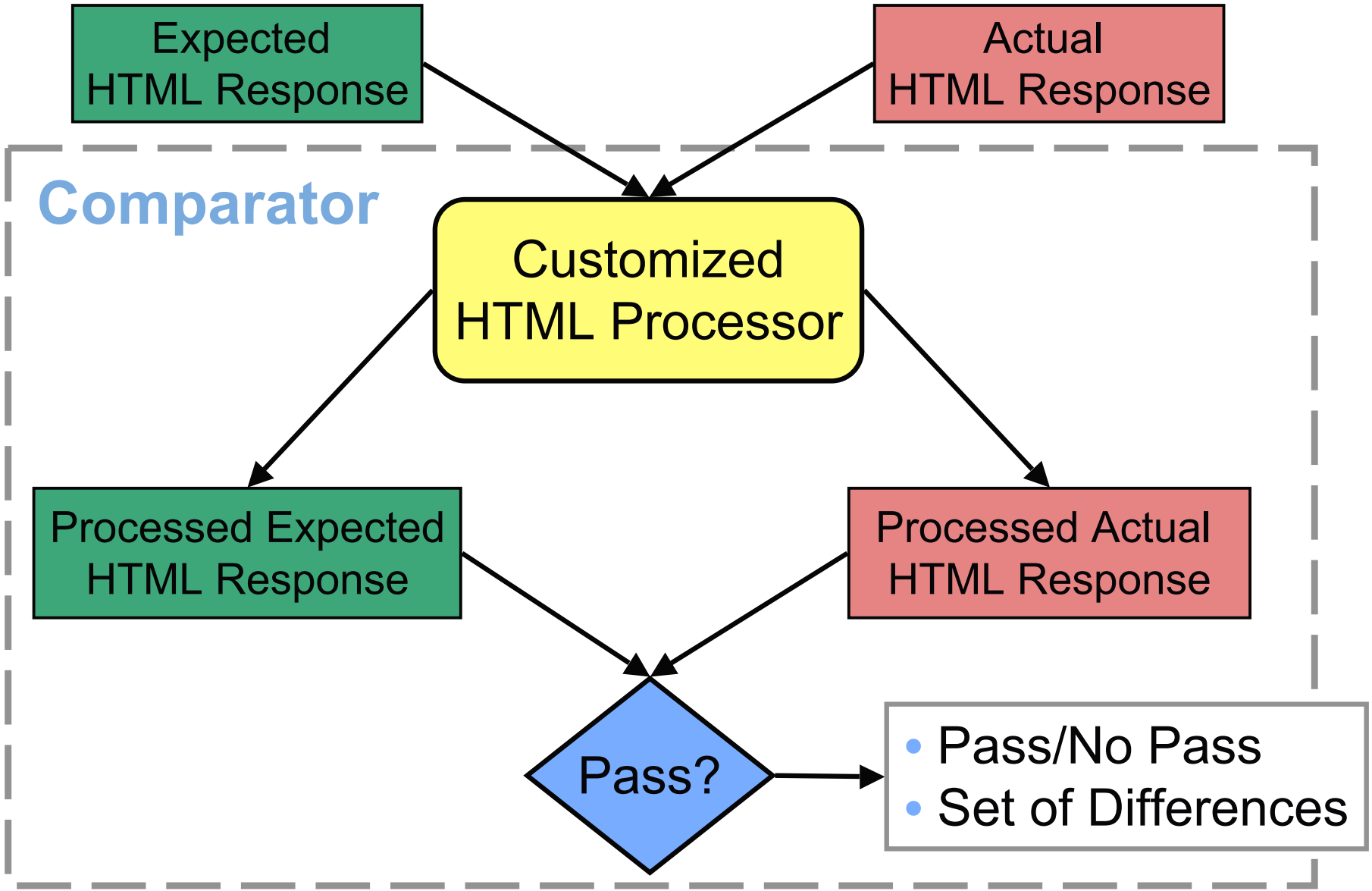
- Developed 22 automated HTML comparison algorithms
  - Why HTML? Common web application output, users *see* output, side effects of failures may show up in HTML
- Evaluated comparators on 4 deployed subject applications
  - False positives, false negatives
- Made recommendations to testers about selecting effective oracles

Our Oracle Process



- Expected results: use original version of application (assumed to be correct)
  - Gold Standard

Our Oracle Process



Importance of Accurate Oracle

- Oracle is quality inspector
- Need oracle to be accurate
  - Allow only the applications that behave as expected to be deployed
- Consequences of inaccurate oracle
  - Buggy application is deployed
    - Customers may see errors
  - Correct application is not deployed
    - Not taking advantage of improvements

Evaluating Oracles

		Application Under Test	
		Faulty Application	Correct Application
Oracle Result (Is there a fault?)	Positive (fault)	True Positive	False Positive Look for a non-existent fault!
	Negative (no fault)	False Negative Missed a fault!	True Negative

- False positive
  - Oracle says “fault” but application does not have a fault
- False negative
  - Oracle says “no fault” but application contains a fault

Evaluating Oracles

		Application Under Test	
		Faulty Application	Correct Application
Oracle Result (Is there a fault?)	Positive (fault)	True Positive	False Positive Look for a non-existent fault!
	Negative (no fault)	False Negative Missed a fault!	True Negative

- HTML comparator algorithms
  - False negatives: for faults that do not show up in the HTML

Example of an HTML Response

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>hiperspace lab</title>
<style>a:hover{color:#952C2C; text-decoration:none}… </style>
<script language="Javascript"> … </script>
</head>
<body>
<table border=0 cellspacing=0 width="580">
<tr> <td rowspace=2>
<img src="books.gif" width=310></td></tr>
…</table>
<!-- Sidebar Links -->
<ul>…
<li><a href="alumni.html">Alumni</a>
</ul>
</body>
</html>
```

Start tag

Content

Style

Layout

Comment

Content

Attribute

Close tag

- Browser collapses white space

Comparing the HTML Output

Expected

```
<html>
<head><title>My Page</title>
</head>
<body>
<h1>Intro</h1>
<p> Text…
<p> Today is January 10.
<a href="link.html">link</a>
<h2>Subsection</h2>
<form method=post …>…</form>
</body>
```

Actual

```
<html>
<head><title>My Page</title>
</head>
<body>
<h1>Intro</h1>
<p> Text…
<br> Today is January 11.
<a href="link2.html">link</a>
<h2>Subsection</h2>
<form method=post …>…</form>
</body>
```

Comparison Algorithm: Document

- Diff entire document
  - Cheap, thorough

```
<html>
<head><title>My Page</title>
</head>
<body>
<h1>Intro</h1>
<p> Text…
<p> Today is January 10.
<a href="link.html">link</a>
<h2>Subsection</h2>
<form method=post …>…</form>
</body>
```

```
<html>
<head><title>My Page</title>
</head>
<body>
<h1>Intro</h1>
<p> Text…
<br> Today is January 11.
<a href="link2.html">link</a>
<h2>Subsection</h2>
<form method=post …>…</form>
</body>
```

Not a fault

Appearance fault

Link fault

Comparison Algorithm: Document

- Diff entire document
  - Cheap, thorough

```
<html>
<head><title>My Page</title>
</head>
<body>
<h1>Intro</h1>
<p> Text…
<p> Today is January 10.
<a href="link.html">link</a>
<h2>Subsection</h2>
<form method=post …>…</form>
</body>
```

```
<html>
<head><title>My Page</title>
</head>
<body>
<h1>Intro</h1>
<p> Text…
<br> Today is January 11.
<a href="link2.html">link</a>
<h2>Subsection</h2>
<form method=post …>…</form>
</body>
```

- **Options**: ignore comments, some tags; collapse white space

Comparison Algorithm: Content

- Diff document's text

My Page Intro Text… Today is January 10. Subsection	My Page Intro Text… Today is January 11. Subsection

- Misses link fault
- **Options**: collapse white space, ignore dates

Comparison Algorithm: Structure

- Diff document's tags

```
<html>
<head><title></title>
</head>
<body>
<h1></h1>
<p>
<p>
<a href="link.html">
<h2></h2>
<form method=post …>…</form>
</body>
```

```
<html>
<head><title></title>
</head>
<body>
<h1></h1>
<p>
<br>
<a href="link2.html">
<h2></h2>
<form method=post …>…</form>
</body>
```

- Miss errors in content
- **Options**: ignore closing tags, style, layout, unimportant attributes, all attributes, form options, order of links/imgs

HTML Comparison Algorithms

False negatives: faults not manifested in HTML

Comparison Algorithm	False Positives	False Negatives
Document	Dynamic, real-time changes	————
Content	Dynamic, real-time changes	Errors in structure, e.g. forms
Structure	Display changes that do not affect app behavior	Errors in content

Partial Ordering of Implemented Comparison Algorithms

Document-based oracles:

Document (D)
DocumentBase (DB)
DocumentBase-CollapsedWS (DB-W)

Content-based oracles:

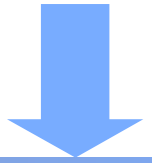
Content (C)
Content-CollapsedWS (C-W)
Content-CollapsedWS-Dates (C-WD)

Structure-based classes:

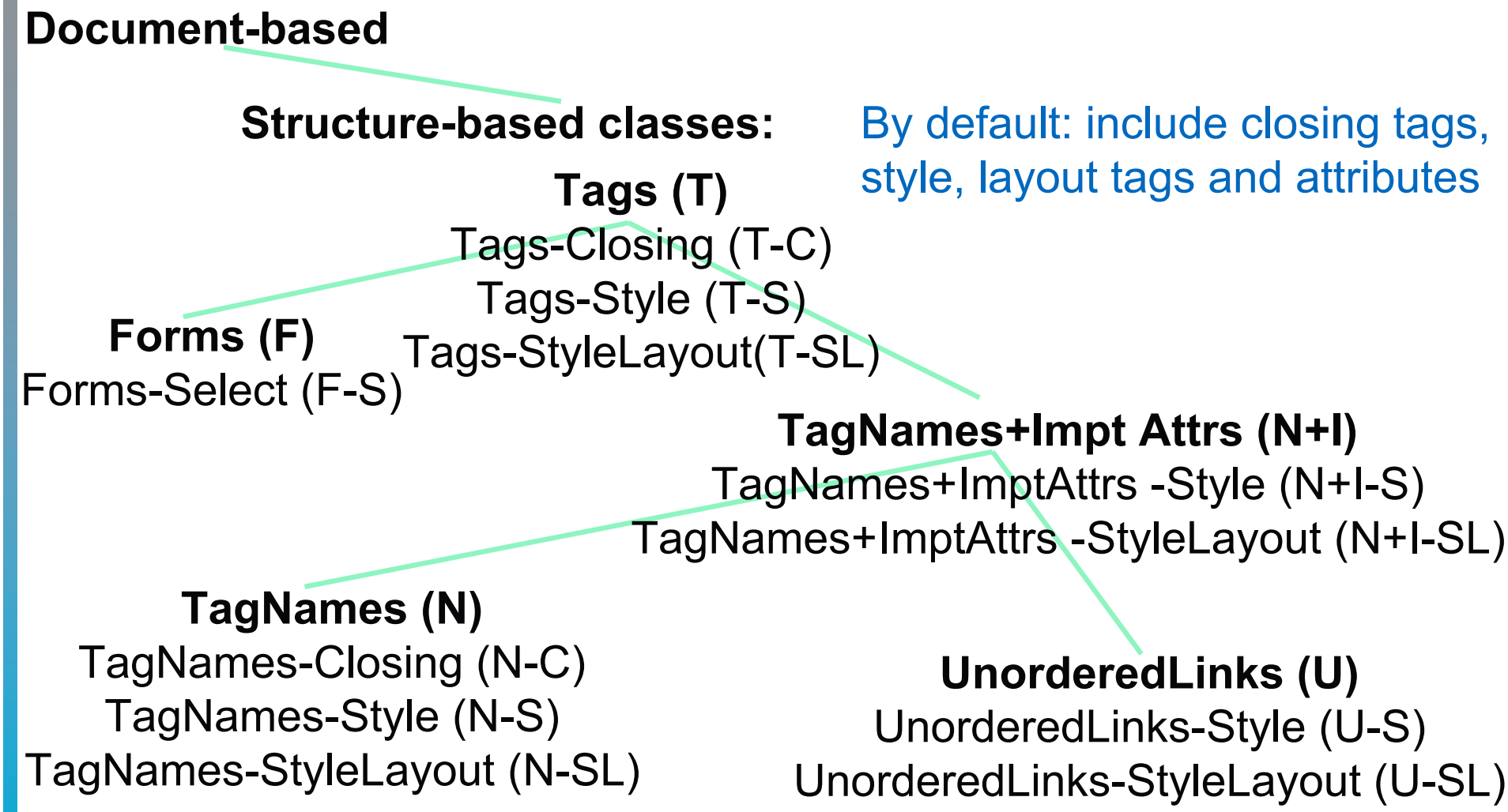
Tags (T)
- Forms (F)
- TagNames+Impt Attrs (N+I)
  - TagNames (N)
  - UnorderedLinks (U)


More information


Fewer False Negatives


Fewer False Positives

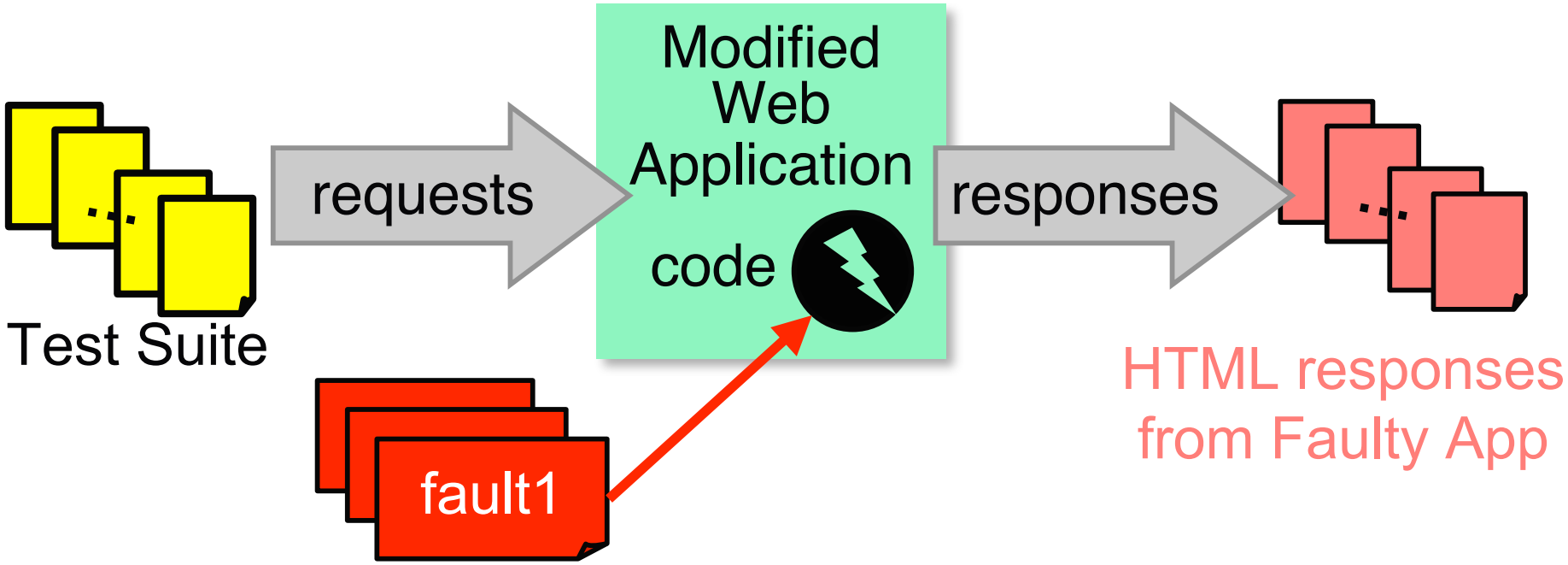
Partial Ordering of Implemented Comparison Algorithms



Experimental Study

- **Goal**: oracle evaluation and recommendations to testers, researchers
- Exp 1: Effect of nondeterministic, real-time behavior
  - False positives
- Exp 2: Failure detection
  - Number & types of detected faults
  - Precision: $\frac{\text{\# failures correct}}{\text{Total failures reported}}$ (higher means few false +)
  - Recall: $\frac{\text{\# failures correct}}{\text{Expected \# of failures}}$ (higher means few false -)

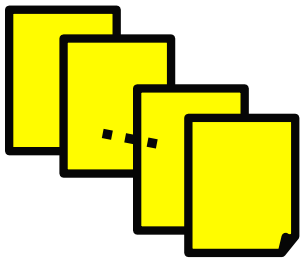
Methodology: Failure Detection Study



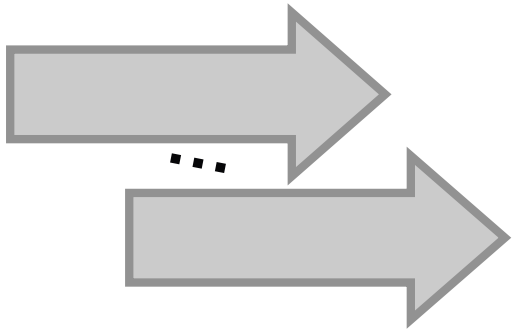
- Faults seeded manually into application code
- Various *categories* of seeded faults
  - data, logic, form, appearance, link

Methodology: Evaluating Oracles

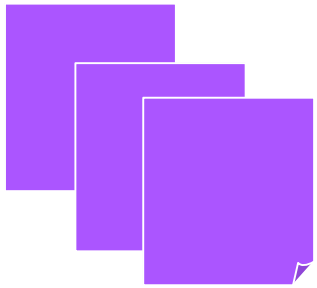
For the test suite of each subject application:



Test Suite



Replay suite on clean and fault-seeded version(s) of code



Generate failure detection reports for each **fault class** using each **oracle**

Comparator's Failure Detection Report:

Request/Response	Failure? (Yes/No)

Subject Applications

- Four deployed subject applications

Application	Test Cases	Requests	NCLOC*	Faults Exposed
Masplas	169	1103	999	22
E-Commerce Bookstore (Book)	125	3564	7791	36
Course Project Manager (CPM)	890	12352	9300	96
DSpace	75	3183 (3023 HTML pages)	49513	20

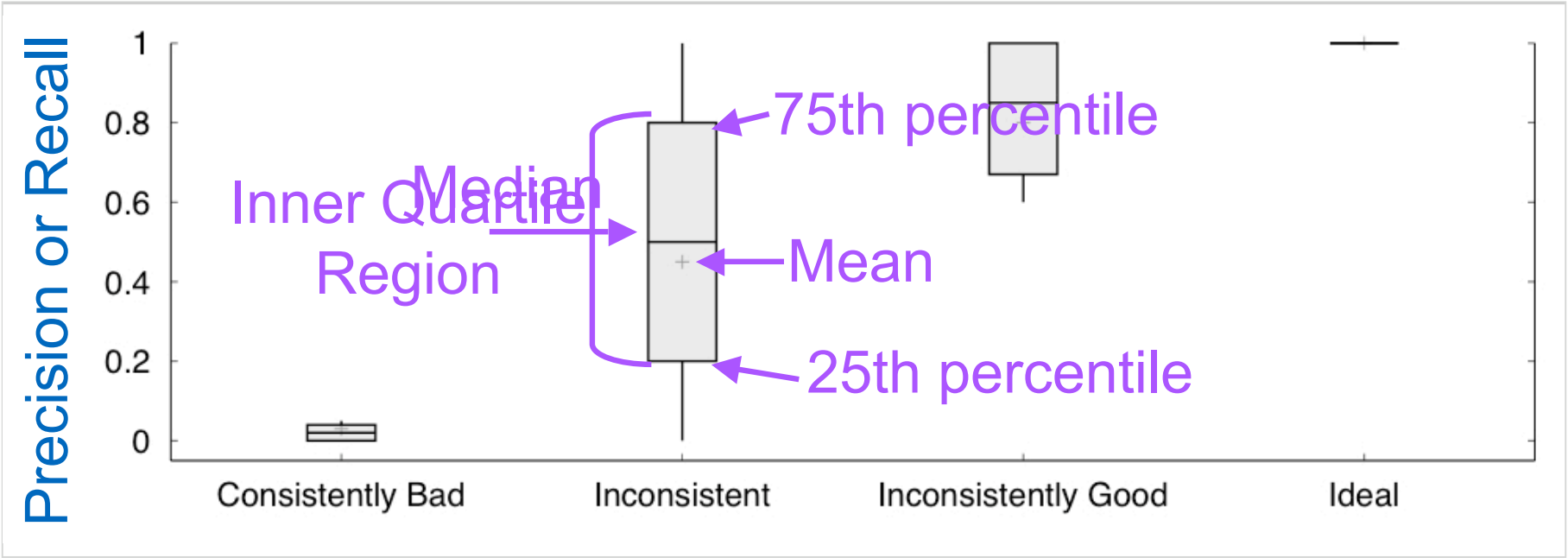
Deterministic behavior

Nondeterministic, real-time behavior

*NCLOC: Non-comment Lines of Code

Total: 174 Faults

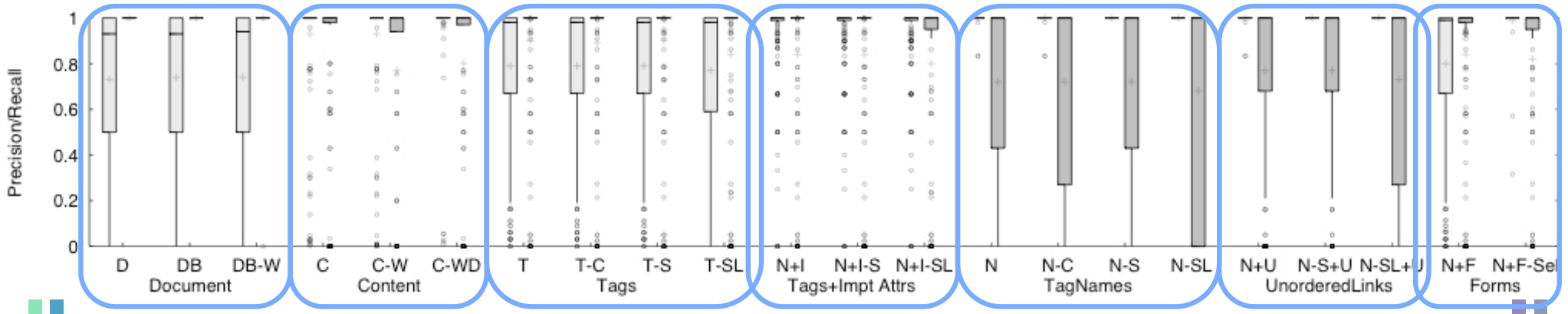
How to Interpret Results



- Legend:
  - Horizontal Bar: Median
  - + : Mean
  - Box: Inner quartile region (IQR) -- inner 50% of data
  - Vertical line: 1.5 * IQR
- Point in distribution: comparator's precision or recall for 1 fault

Failure Detection Results

- Precision: light gray, left
- Recall: dark gray, right
- Across all applications, all fault types
  - Individual graphs in Sprenkle thesis



- Tradeoff between precision and recall
- Trends follow hierarchy
  - Top of hierarchy: higher recall, lower precision
  - Leaves of hierarchy: lower recall, higher precision

Combinations of Oracles

- **Key Insight**: combine best oracles that focus on disparate parts of the HTML document to improve effectiveness over individual comparators
  - Union: improves recall, may affect precision
  - Intersection: improves precision, decreases recall
- Empirically best combination, across all apps: **C-WD ∪ N+I**

Guidance to Testers

- Execute test suite several times
  - Identify responses with nondeterministic, real-time behavior
  - Identify comparators with fewest false positives
- Vary comparator by response's behavior
  - Deterministic: use Document
  - Nondeterministic: use C-WD ∪ N+I
- Use more precise oracle in early testing stages
  - Don't overwhelm testers with failure reports
  - Flesh out bugs w/o sifting through false positives
- **Document** family for fewest false negatives
- **Forms-Select** for fewest false positives w/ highest recall

Related Work

- Model-based testing of web applications
  - Ricca01, Liu00, Deng04
  - Andrews05: suggests partial oracles
- HttpUnit
  - Testers create oracles manually
- HTML-based comparators
  - Elbaum05, DiLucca02 -- few details of implementation, no evaluation of false positives, negatives

Conclusions and Future Work

- Proposed, evaluated a suite of customized, HTML-based oracle comparators
- Recommendations to testers for selecting oracle
  - Most effective oracle in general: Content-CollapsedWS-Dates ∪ TagNames+ImptAttrs
- Learning effective oracle comparator combinations: STEV2007
  - Train on faults from bug reports
- Future work
  - Fully customizable comparator
  - Evaluate with more applications, faults