

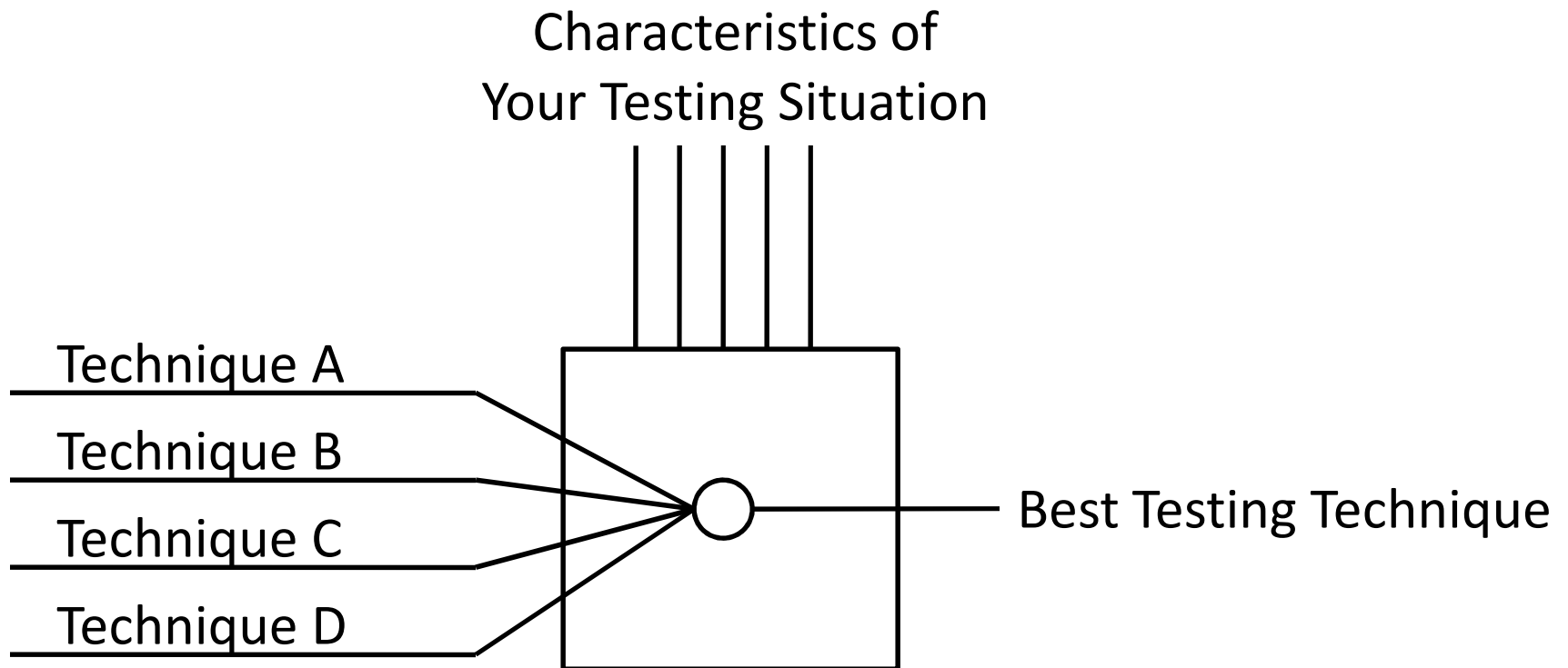
Relationships Between Test Suites, Faults, and Fault Detection in GUI Testing



Jaymie Strecker
University of Maryland, College Park
strecker@cs.umd.edu
ICST 2008

Joint work with Atif Memon, University of Maryland

The Holy Grail



The Holy Grail (2)

- Which characteristics of testing situations affect testing outcomes?
 - Faults
 - Test suites
 - Applications
 - ...
- How do these characteristics affect outcomes?
 - Fault detection
 - Cost
 - ...
- Given my situation, what testing technique should I use?

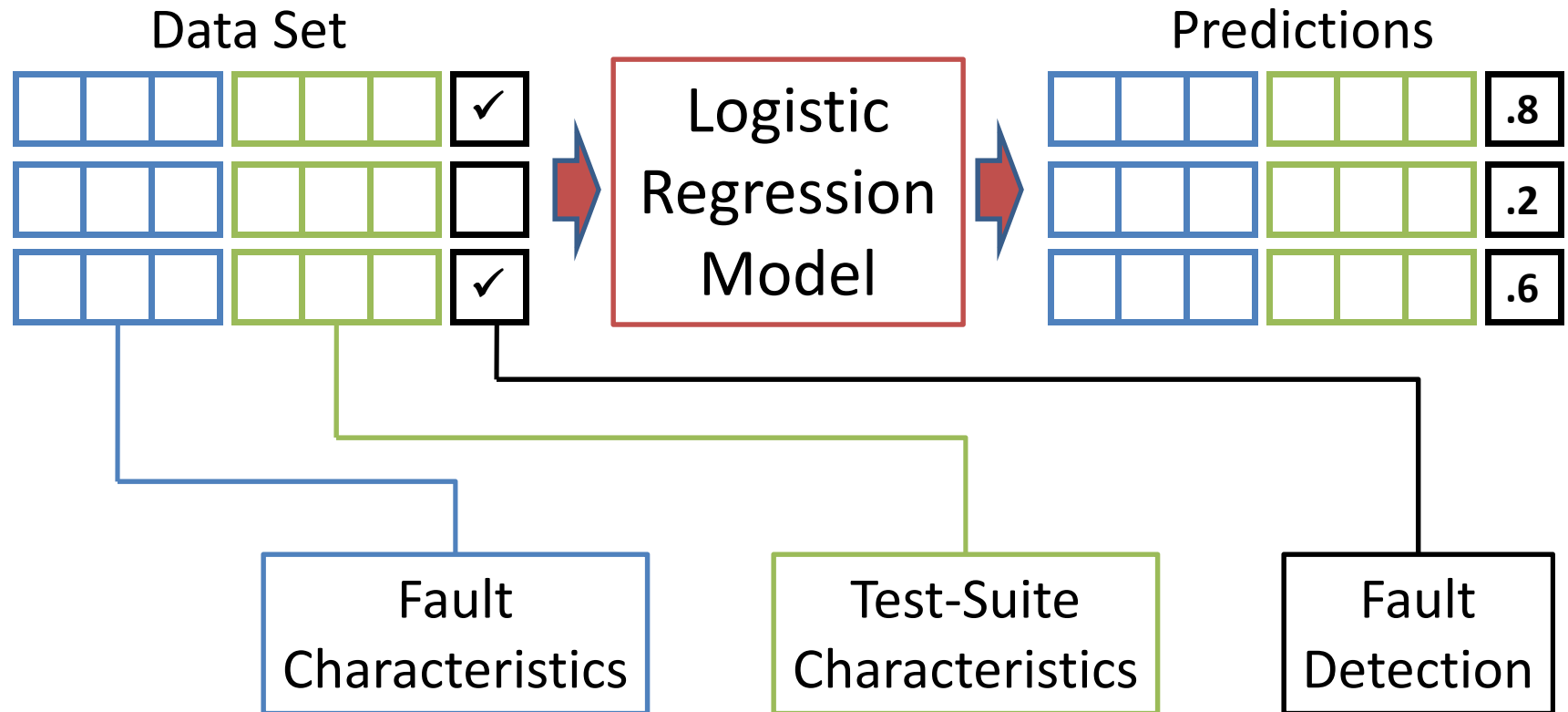
Why This Is Hard

- Experimenting
 - Diverse applications, test suites, faults, etc.
 - Technical infrastructure
 - Cultural infrastructure
 - Progress: Software-artifact infrastructure repository (<http://sir.unl.edu>), e.g.
- Characterizing testing situations
 - Especially faults
 - Progress: Elbaum et. al (METRICS'01), Rothermel et. al (TOSEM'04), Offutt et. al (ISSTA'96), e.g.

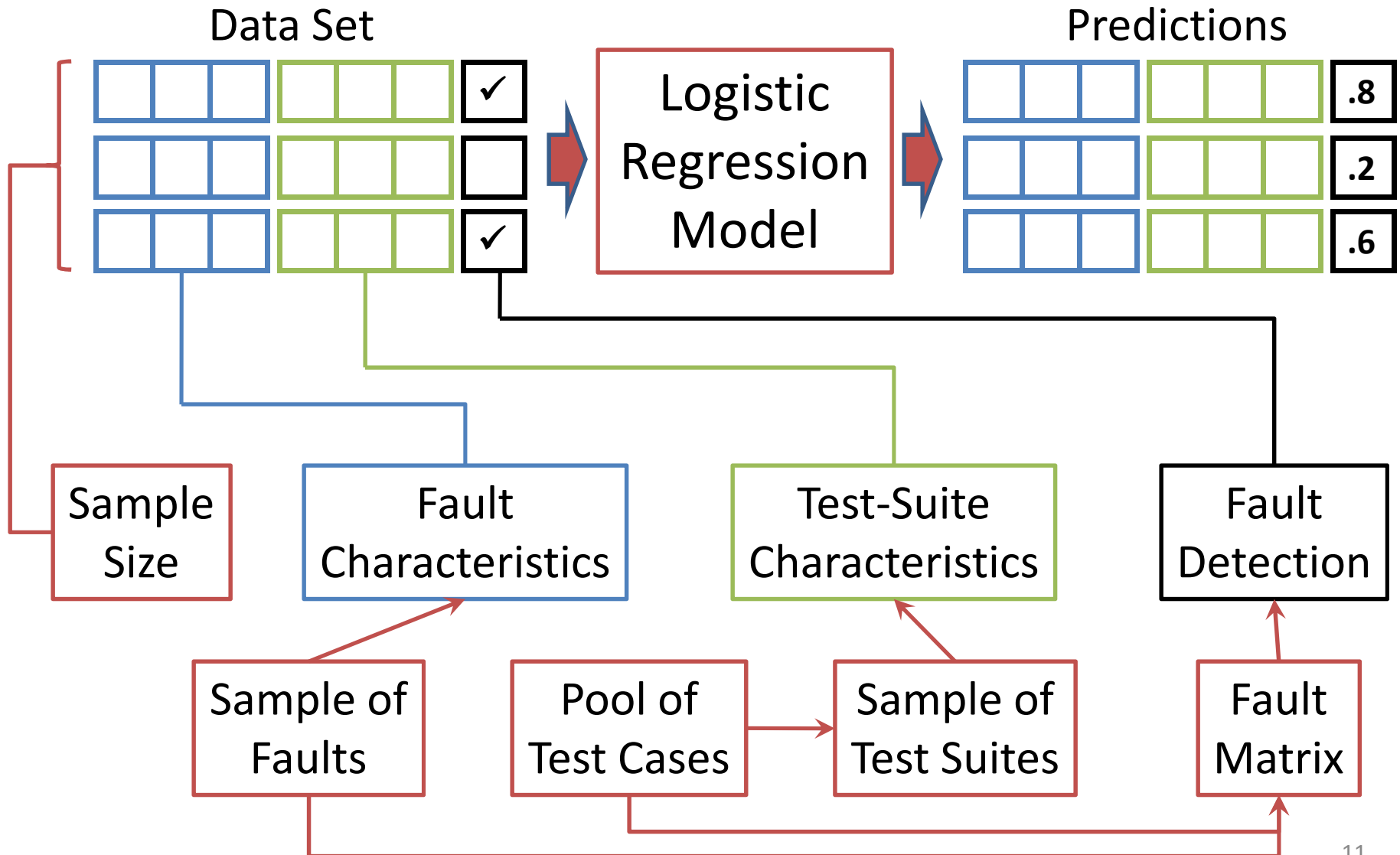
A New Methodology

- Process for studying relationship between testing situations and testing outcomes
 - Testing situations: test suites and faults
 - Testing outcomes: fault detection
- Reusable
- Automated
- Step toward “holy grail”

Methodology



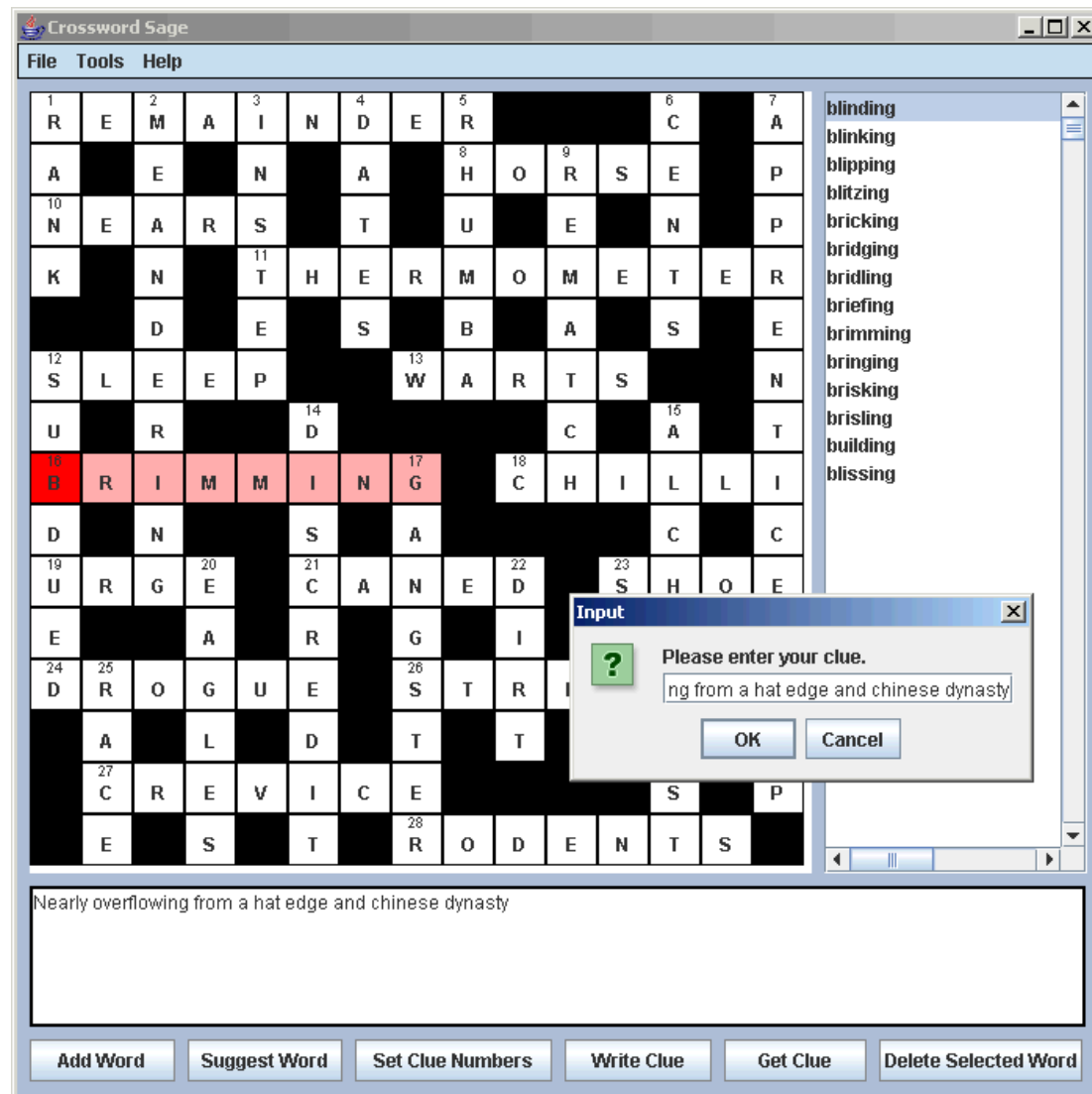
Methodology



Contributions

- Methodology for studying relationship between testing situations and testing outcomes
- Experiment that uses methodology
 - Study effects of test-suite and fault characteristics on fault detection
 - 2 real, GUI-based applications

GUI Testing



GUI Testing

Widgets



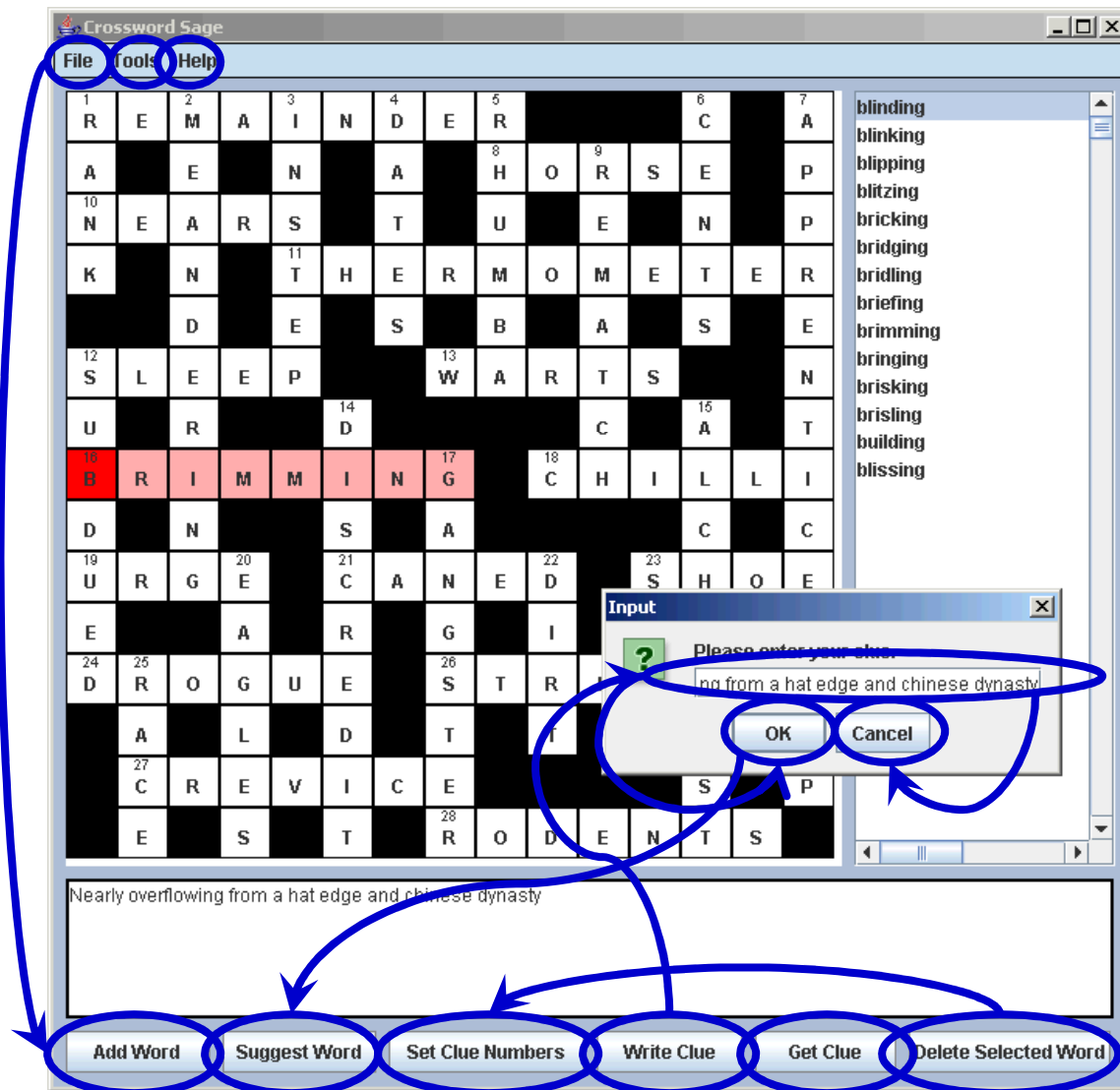
GUI Testing

Events

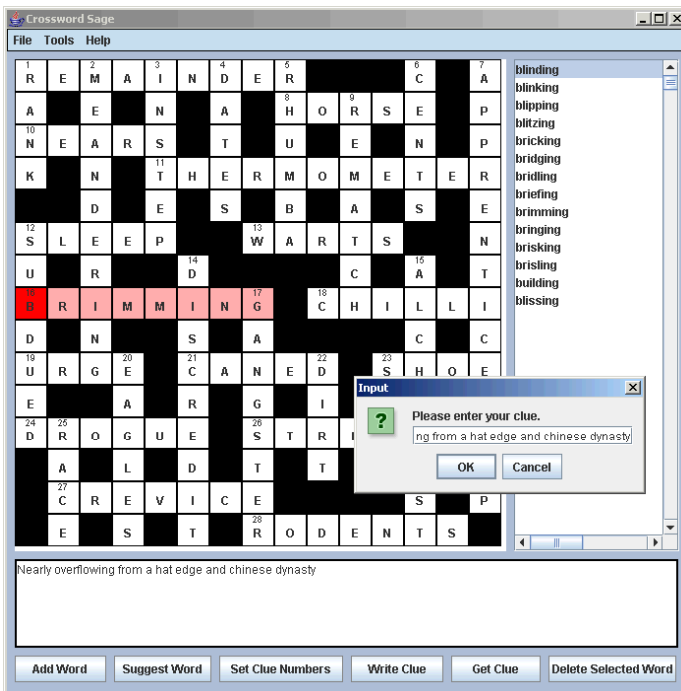


GUI Testing

Test Cases

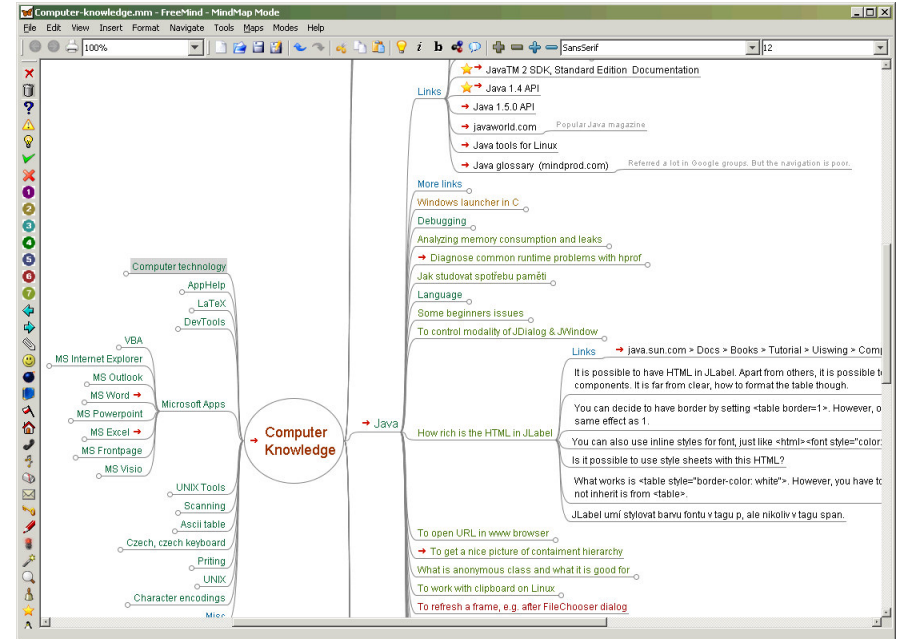


Applications Studied



CrosswordSage

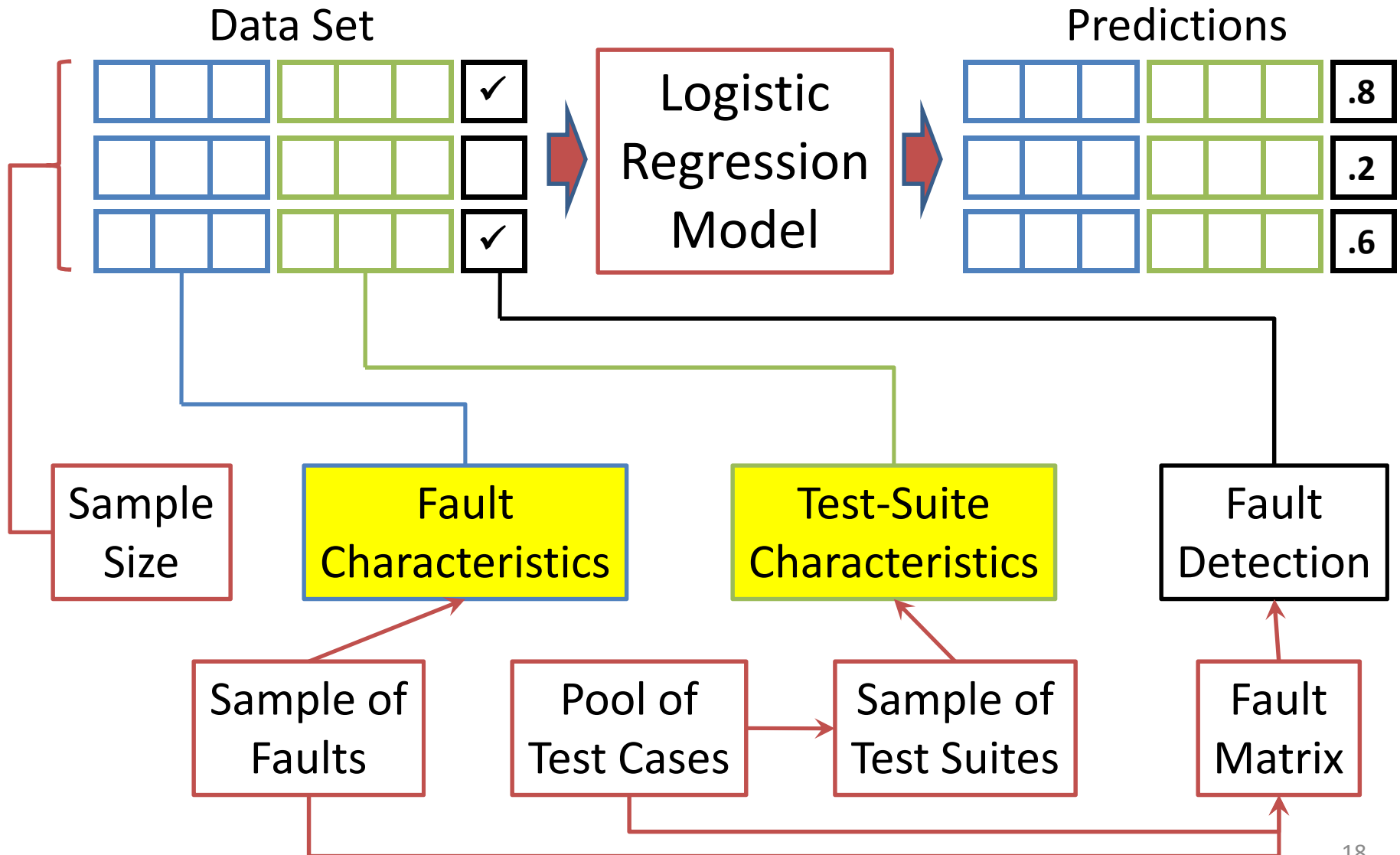
- 3220 LOC
- 36 classes
- crosswordsage.sourceforge.net



FreeMind

- 24665 LOC
- 858 classes
- freemind.sourceforge.net

Instantiated Methodology



Experiment Variables

- **Independent variables:** characteristics of test suites and faults that may affect fault detection

Test Suites

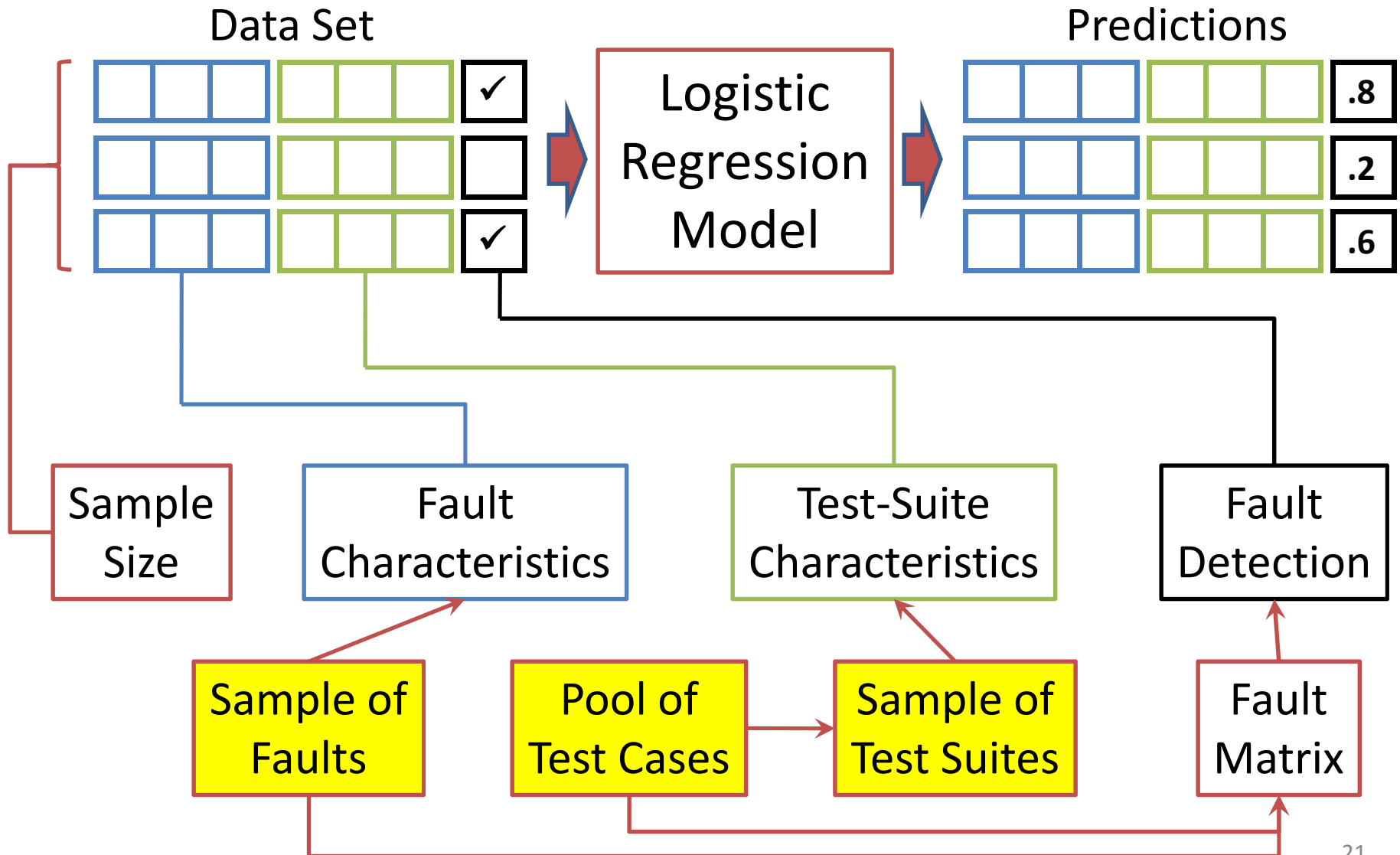
Len	Length of test cases
Size	Size (num. of events)
E2Cov	Event-pair coverage / size
E3Cov	Event-triple coverage / event-pair coverage

Faults

Mut	Mutant type (method- or class-level)
Branch	Branch points in faulty method's bytecode
StmtDet	Estimated probability of detection by statement-coverage-adequate test suites

- **Dependent variable:** fault detection in $\langle test\ suite, fault \rangle$ pairs (Det = 0 or 1)

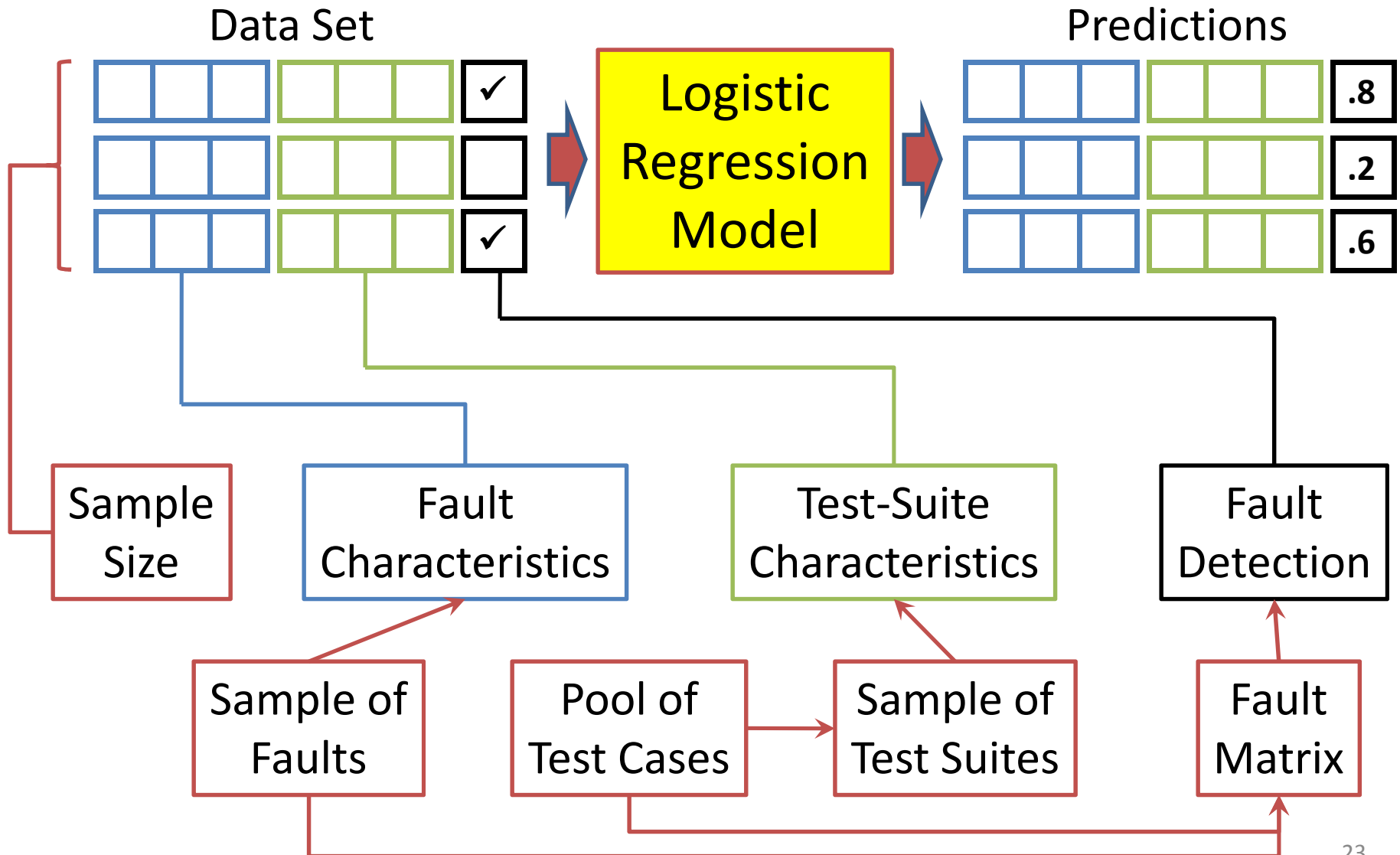
Instantiated Methodology



Test Suites and Faults Studied

- **Faults**: random sample of mutants generated by MuJava
 - Method-level: inserting decrement operator at variable use, e.g.
 - Class-level: changing type of data member, e.g.
- **Test pool**: semi-random sample of GUI test cases
 - Pool covers each event at least x times
- **Test suites**: random sample of subsets of test pool
 - Each suite covers all events
 - All test cases in a suite have same length

Instantiated Methodology



Data Analysis

- Logistic regression models

$$\log\left(\frac{P}{1-P}\right) = \alpha + \vec{\beta} \cdot \vec{I}$$

Probability that dependent variable = 1

Independent variables

Coefficients

Ex: $\log\left(\frac{P}{1-P}\right) = -13.8 + 0.6Mut + 7.5StmtDet + 7.6E3Cov$

- **Full model:** Start with all independent variables
- **Reduced model:** Remove one at a time in stepwise regression

Results: CrosswordSage

Full Model

Term	Coef.	Odds Factor (10%)
Mut	0.845	2.328*
StmtDet	8.606	2.365
Branch	-0.006	0.999
E3Cov	10.124	2.752
E2Cov	1.575	1.171
Size	-0.010	0.999
Len	0.302	1.030

Reduced Model

Term	Coef.	Odds Factor (10%)
Mut	0.648	1.912*
StmtDet	7.510	2.119
E3Cov	7.630	2.145

* Odds factor for method-level vs. class-level mutants

Terms in **bold** are statistically significant at the 0.05 level.

Results: FreeMind

Full Model

Term	Coef.	Odds Factor (10%)
Mut	-1.872	0.154*
StmtDet	8.287	2.290
Branch	-0.878	0.916
E3Cov	-11.276	0.324
E2Cov	76.815	2167.869
Size	-0.009	0.999
Len	0.613	1.063

Reduced Model

Term	Coef.	Odds Factor (10%)
StmtDet	6.094	1.839

* Odds factor for method-level vs. class-level mutants

Terms in **bold** are statistically significant at the 0.05 level.

Conclusions: Experiment

- StmtDet important for both applications
- Mut and E3Cov important for CrosswordSage only
- Can plug values for independent variables into logistic regression models to predict fault detection
- Future work
 - More/better test-suite and fault characteristics
 - Replicate, replicate, replicate!

Conclusions: Methodology

- Successfully used in our experiment
- Main threat to validity: test pool
 - Necessary for 146 test suites composed of up to 424 test cases each
- Makes experiment replication easier
 - Automation
- Leads to results you can use
 - Statistical tests
 - Predictive models
 - Helps you find the best testing technique for your situation

Thank You!

- Software-testing benchmarks
 - <http://www.cs.umd.edu/~atif/Benchmarks/UMD2007b.html>
(this study's applications, test suites, faults, and tools)
 - <http://www.cs.umd.edu/~atif/Benchmarks/> (additional benchmarks)
- Contact information
 - Jaymie Strecker: strecker@cs.umd.edu
 - <http://www.cs.umd.edu/~strecker/>

Example Study

Mean % of faults detected

	Code Reading	Functional Testing
Control Faults	42.8	66.7
Interface Faults	46.7	30.7
Total	54.1	54.6

V. R. Basili and R. W. Selby. Comparing the effectiveness of software testing strategies. *IEEE Trans. Softw. Eng.*, 13(12):1278-1296, 1987.