Visualization of Test Information to Assist Fault Localization

James A. Jones, Mary Jean Harrold, John Stasko

Georgia Institute of Technology

Supported by Boeing Aerospace Company, National Science Foundation, and the Yamacraw Project

Aristotle Research Group

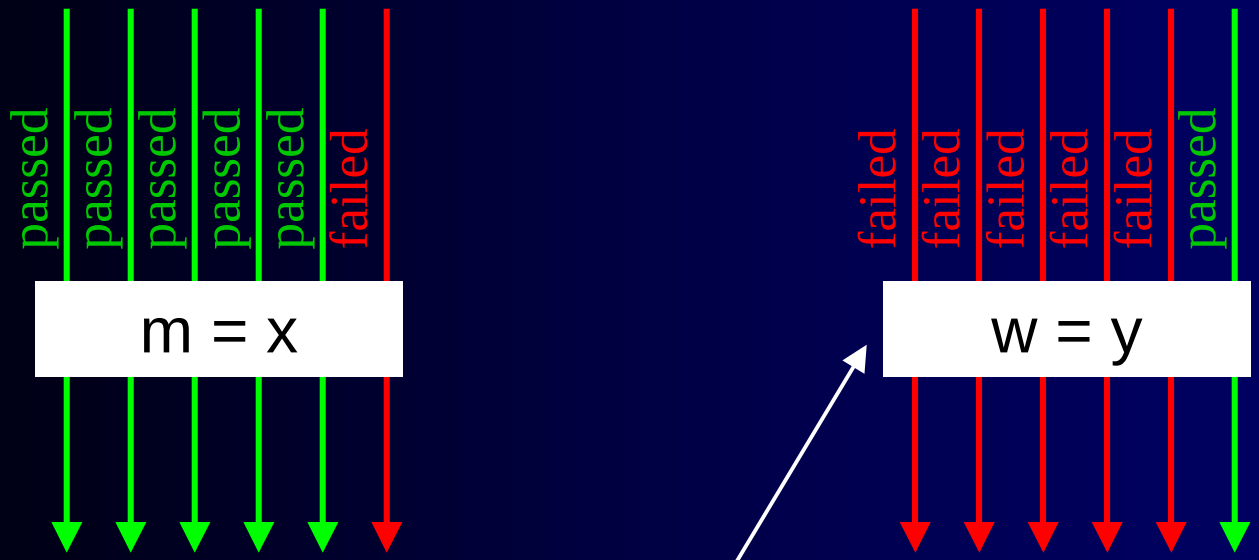
Information Interfaces

Outline

- Approach
- Demonstration of Prototype
- Case studies
- Related Work
- Conclusions & Future Work

Approach

Consider two statements



Approach

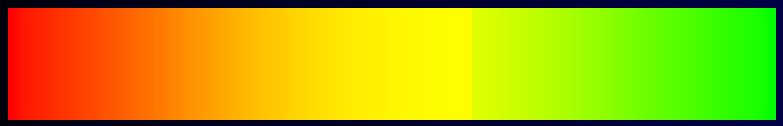
- Utilizes
  - pass/fail results of executing test cases
  - coverage provided by those test cases
  - source code of program under test
- Provides visualization of program statements that summarizes pass/fail status of test cases that covered them

Approach

For statement *s*:

Hue summarizes pass/fail results of test cases that executed *s*

Brightness presents the "confidence" of the hue assigned to *s*



Hue

- Summarizes pass/fail results
  - executed mostly by failed test cases → red
  - executed evenly by passed and failed → yellow
  - executed mostly by passed test cases → green
- Uses percentages of passed and failed test cases

$\frac{10}{10}$ failed test cases = 100%

```
m = y;
```

$\frac{20}{200}$ passed test cases = 10 %

$$\text{hue(s)} = \text{low hue (red)} + \frac{\%\text{passed(s)}}{\%\text{passed(s)} + \%\text{failed(s)}} * \text{hue range}$$

Hue

$\frac{10}{10}$ failed test cases = 100%

$\frac{0}{10}$ passed test cases = 0%

```
a = b;
```

$\frac{1}{10}$ failed test cases = 10%

$\frac{0}{10}$ passed test cases = 0%

```
c = d;
```

$\frac{10}{10}$ failed test cases = 100%

```
m = y;
```

$\frac{20}{200}$ passed test cases = 10 %

$$\text{hue(s)} = \text{low hue (red)} + \frac{\%\text{passed(s)}}{\%\text{passed(s)}+\%\text{failed(s)}} * \text{hue range}$$

Aristotle Research Group

Information Interfaces

Brightness

- Shows the amount of coverage for a statement
- Uses greater percentage of passed and failed to show confidence in hue assigned

$\frac{10}{10}$ failed test cases = 100%

```
m = y;
```

$\frac{20}{200}$ passed test cases = 10 %

$$\text{bright}(s) = \max(\%\ \text{passed}(s), \%\ \text{failed}(s))$$

Example

```
   mid() {
    int x,y,z,m;
1:  read("Enter 3 numbers:",x,y,z);
2:  m = z;
3:  if (y<z)
4:     if (x<y)
5:           m = y;
6:     else if (x<z)
7:           m = y;
8:  else
9:     if (x>y)
10:          m = y;
11:    else if (x>z)
12:          m = x;
13: print("Middle number is:", m);
   }
```

	Test Cases					
	3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
1	•	•	•	•	•	•
2	•	•	•	•	•	•
3	•	•	•	•	•	•
4	•	•			•	•
5		•				
6	•				•	•
7	•					•
8			•	•		
9			•	•		
10			•			
11				•		
12						
13	•	•	•	•	•	•
Pass Status:	P	P	P	P	P	F

Aristotle Research Group

Information Interfaces

Example

```
   mid() {
    int x,y,z,m;
1:  read("Enter 3 numbers:",x,y,z);
2:  m = z;
3:  if (y<z)
4:     if (x<y)
5:        m = y;
6:     else if (x<z)
7:        m = y;
8:  else
9:     if (x>y)
10:       m = y;
11:    else if (x>z)
12:       m = x;
13: print("Middle number is:", m);
   }
```

Test Cases	3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
1:	•	•	•	•	•	•
2:	•	•	•	•	•	•
3:	•	•	•	•	•	•
4:	•	•			•	•
5:		•				
6:	•				•	•
7:	•					•
8:			•	•		
9:			•	•		
10:			•			
11:				•		
12:						
13:	•	•	•	•	•	•
Pass Status:	P	P	P	P	P	F

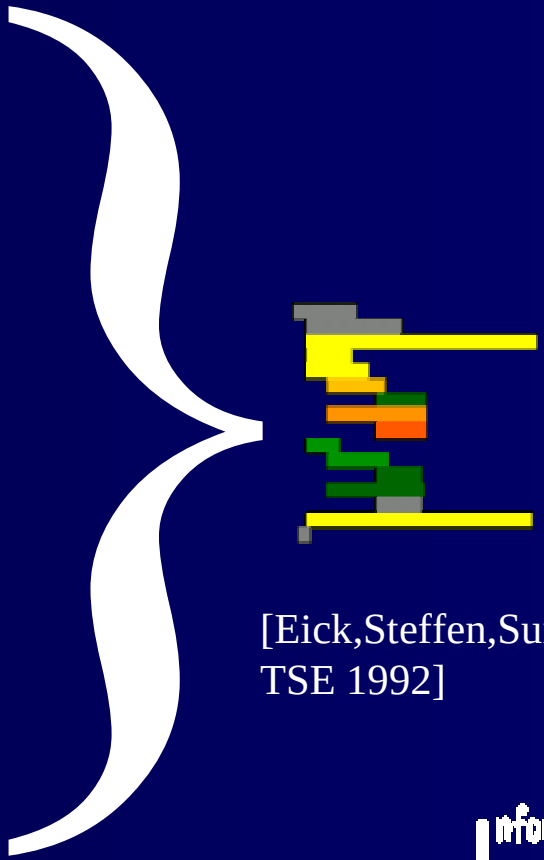
Aristotle Research Group

Information Interfaces

Scalability

- Large programs difficult to display
- Use the line-of-pixels, SeeSoft, view
- Each character in the source is displayed as a pixel

```
mid() {
 int x,y,z,m;
 read("Enter 3 numbers:",x,y,z);
 m = z;
 if (y<z)
    if (x<y)
           m = y;
    else if (x<z)
           m = y;
 else
    if (x>y)
           m = y;
    else if (x>z)
           m = x;
 print("Middle number is:", m);
}
```

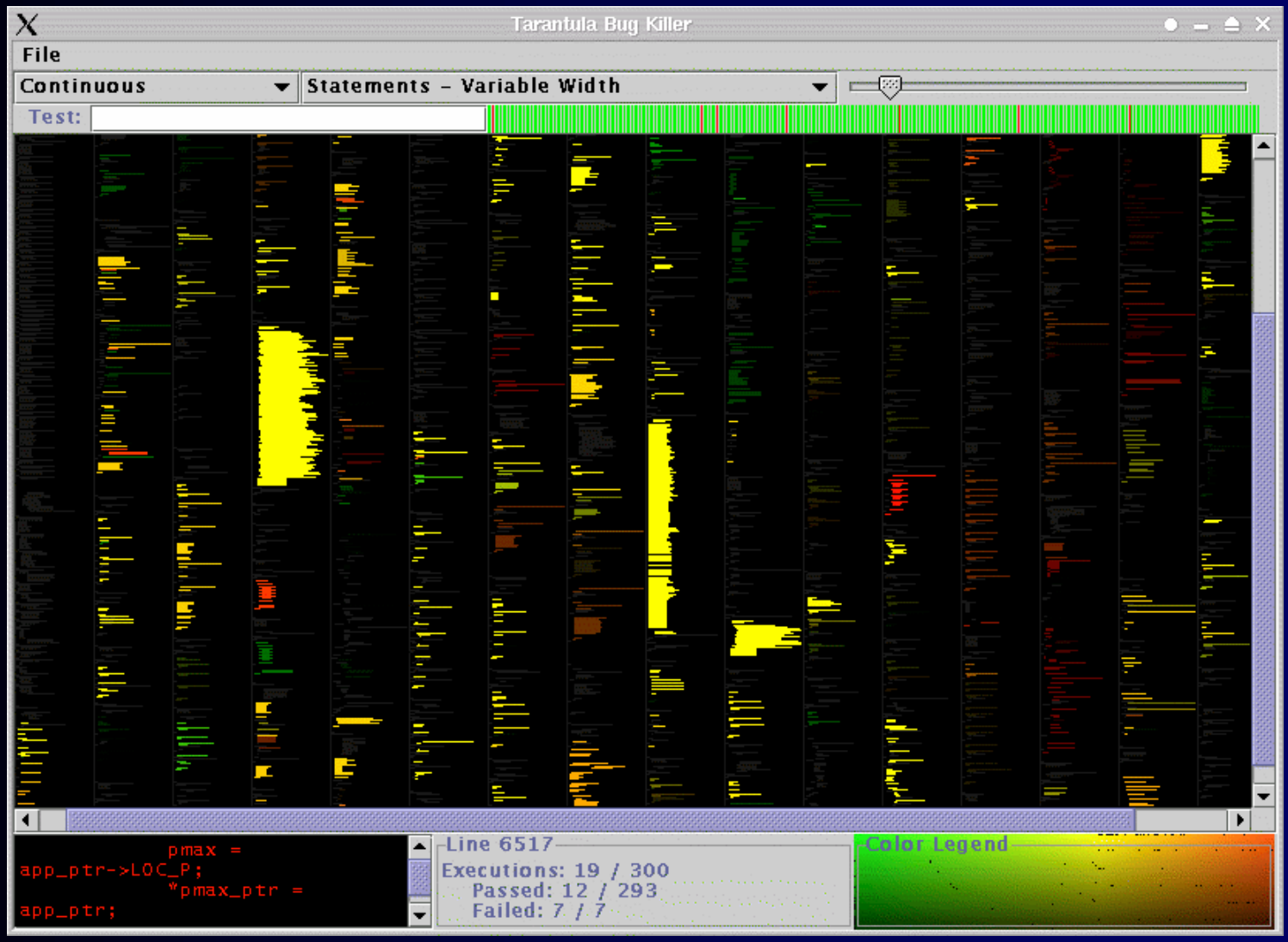


[Eick,Steffen,Sumner, TSE 1992]

Aristotle Research Group

Information Interfaces

Tarantula

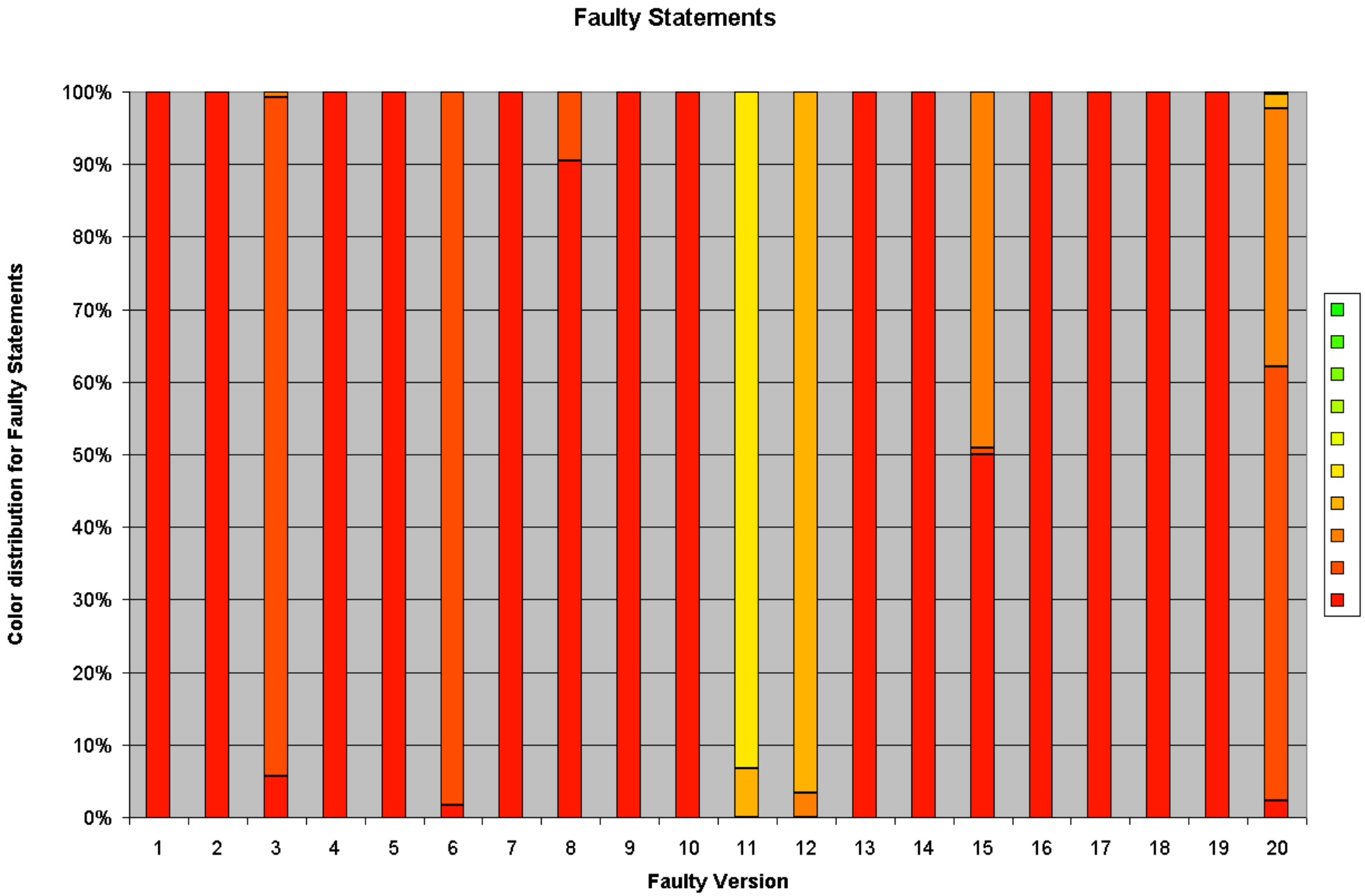


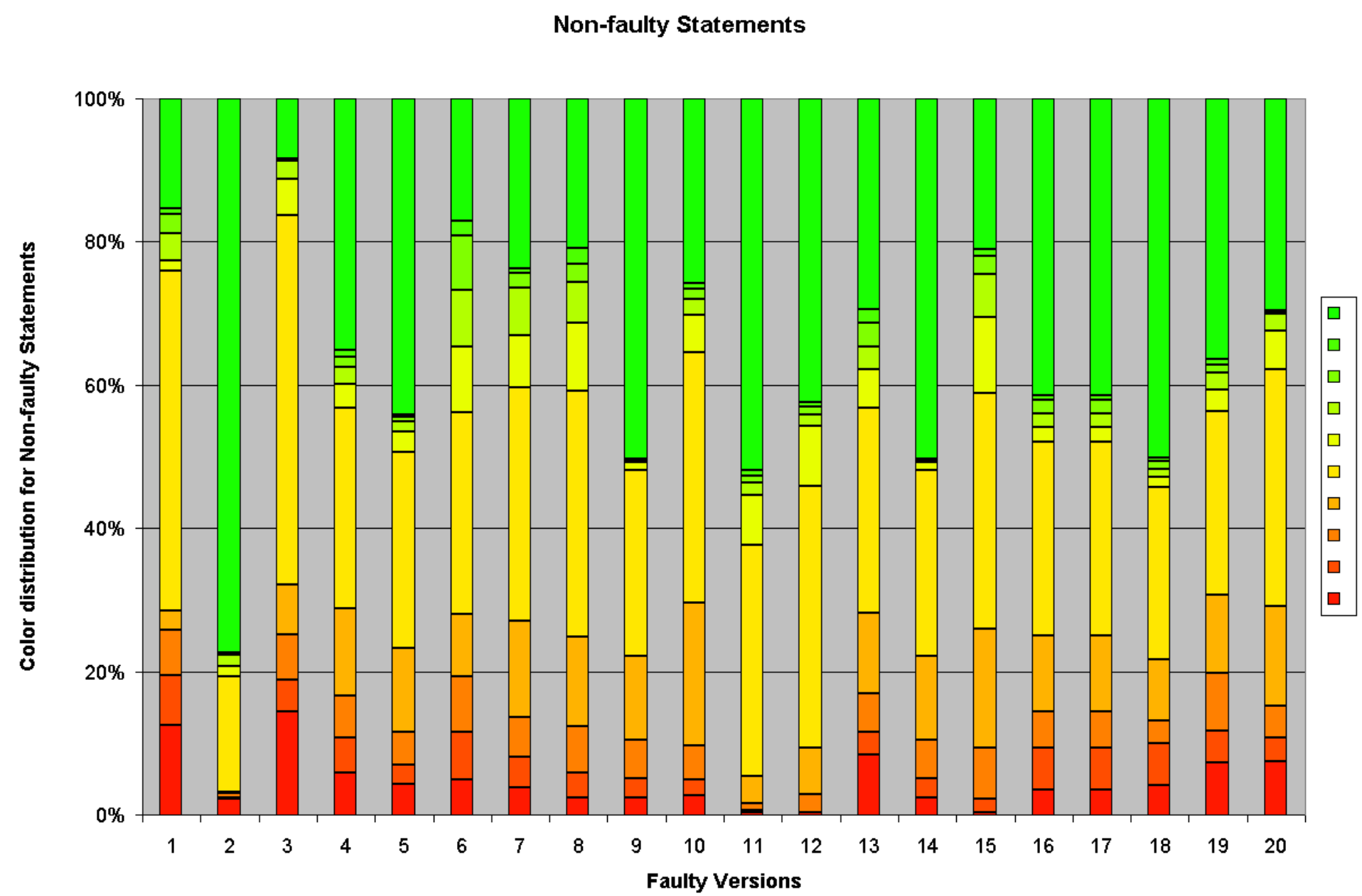
Aristotle Research Group

Information Interfaces

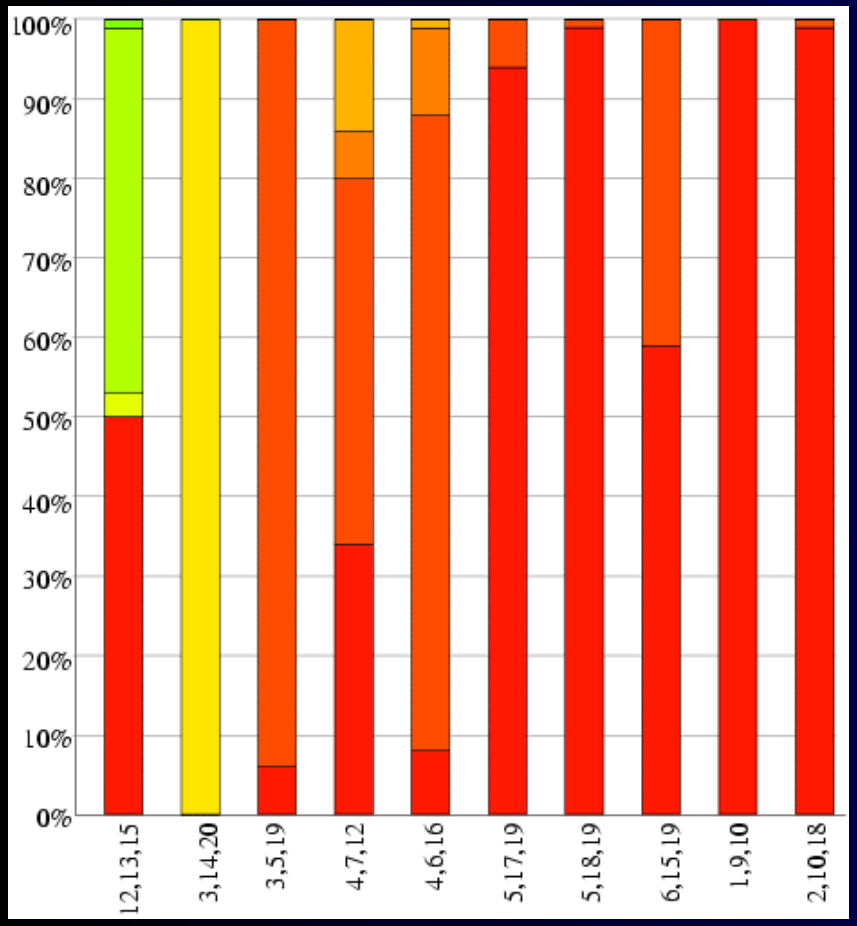
Case Studies

- Two initial studies
  - How red are the faulty statements?
  - How red are the non-faulty statements?
- Subject program: Space
  - 8000 lines of executable code
  - 1000 coverage-based test suites of size 156-4700 test cases
  - 20 faulty versions

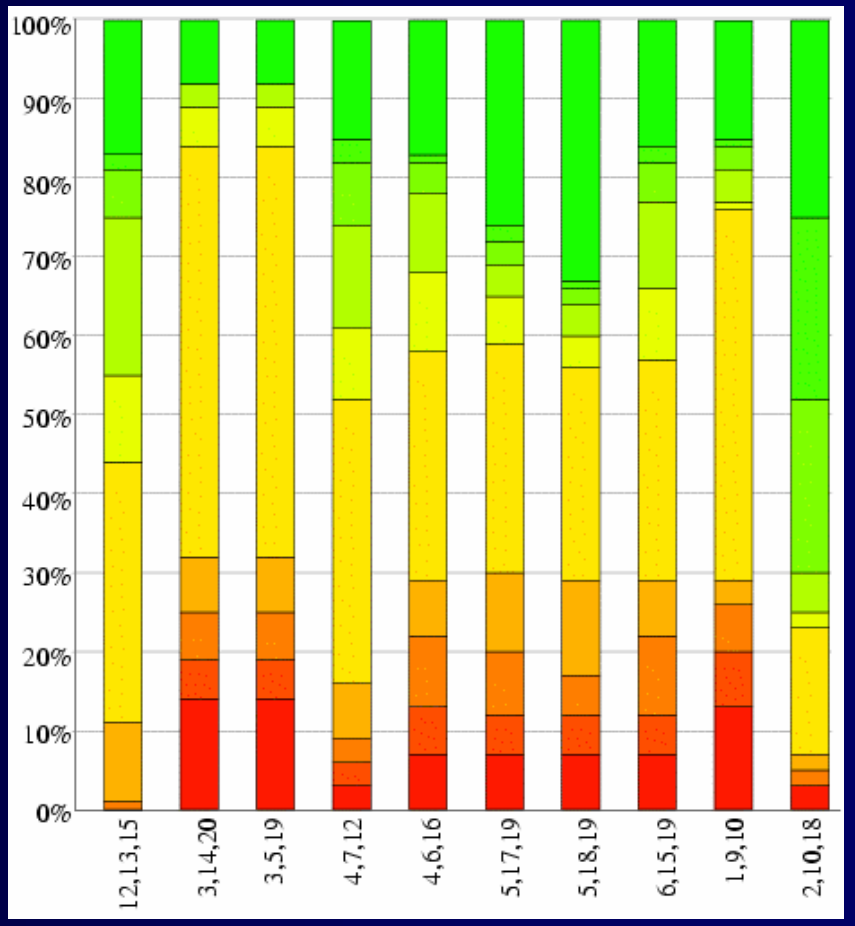

Faulty Statements
Color distribution for Faulty Statements
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Faulty Version


Non-faulty Statements
Color distribution for Non-faulty Statements
100%
80%
60%
40%
20%
0%
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Faulty Versions

Three Faults



Faulty statements



Non-faulty statements

Aristotle Research Group

Information Interfaces

Related Work

- Eick, et al. [TSE92]: SeeSoft visualizes coverage and slices
- Agrawal, Horgan, et al. [ISSRE95]: uses set arithmetic to compute dice for fault localization
- Leon, Podgurski, et al. [ICSE00]: visualize test case behavior using multivariate analysis

Conclusion & Future Work

- New technique that efficiently narrows search space for faults using commonly available information
- Promising results from studies

- Perform more empirical studies
- Create techniques to help when no statements are red
- Provide editing and dynamic update capabilities