Visually Encoding Program Test Information to Find Faults in Software

James Eagan, Mary Jean Harrold,
James A. Jones, John T. Stasko

College of Computing / GVU Center
Georgia Institute of Technology

Supported by Boeing Commercial Airplane Group,
National Science Foundation, and the Yamacraw Project

Aristotle Research Group

Information Interfaces

Area

- Software creation & maintenance
- Locate faults after failures (debugging)
- Reduce the time and cost necessary to debug

Particular Focus

- Testing of large software systems
- Large suites of tests
  - Able to characterize whether a program
    - Passes on a test (Execution is judged correct)
    - Fails on a test (Execution is judged incorrect)

Test Data

- Execution summary on test suite
  - For each test case
    - Its pass/fail status
    - Statements that it executes

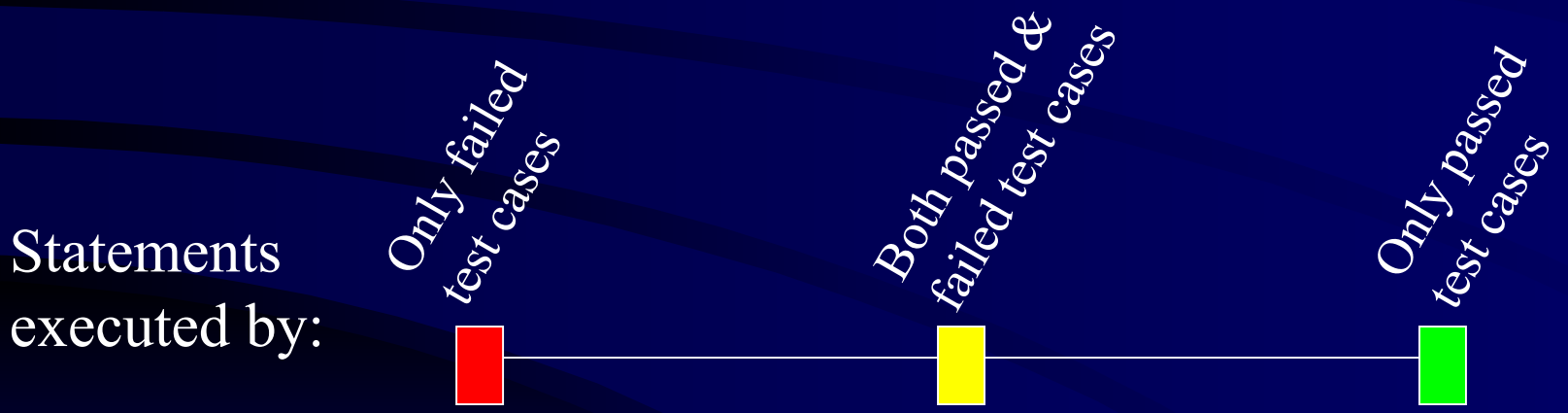
Test #	Status	Statements
1	P	12 13 14 24 25 27 28 …
2	P	12 13 14 15 16 …
3	F	12 13 124 125 126 …
...		

The Problem

- Analyzing one failed test (traditional)
  - Just one data point
  - Neglects context of all other tests
- Analyzing whole suite
  - Huge data set
  - Locating bug by analyzing this textual summary is at best tedious and difficult
  - Desire a better way of representing the summary of the test suite execution

One Approach

- Visualize the execution of all tests
- Display statements in program according to the test cases that execute them



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

Fault

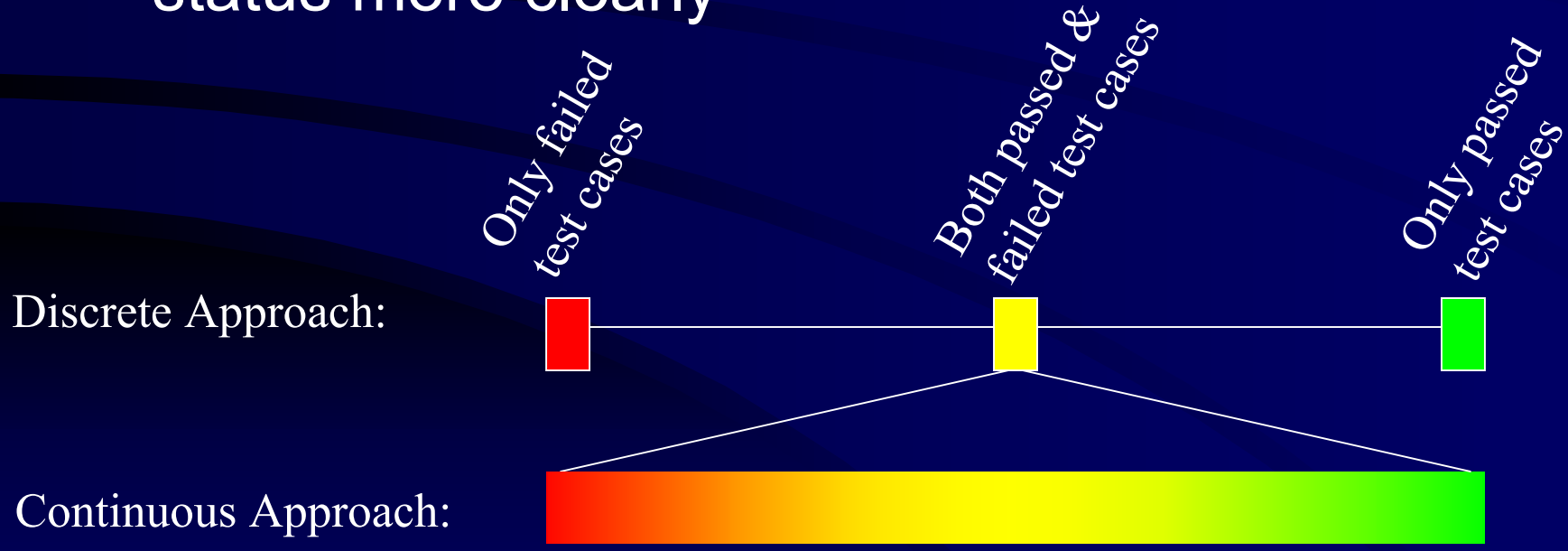
Example

```
    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

Our Approach

- Distribute statements executed by both passed and failed test cases over spectrum
- Use hue and brightness to communicate status more clearly



Hue

- Compare ratios of passed and failed tests through statement
- Slide toward higher percentage

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

Brightness

- Encodes ratio of passed or failed (whichever is higher) through statement
- Higher percentage makes statement brighter, lower makes it darker

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

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

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]

Tarantula

Aristotle Research Group

Information Interfaces

Initial Evaluation

- Two questions to ask
  - How red are the faulty statements?
  - How red are the non-faulty statements?
- Preliminary tests on one system appear promising
  - Faults are typically red
  - Non-faults aren't red very much
  - Submitted ICSE paper

Future Work

- Further evaluation
- Examining perception of colors better
- Understanding bug-finding process
  - What other views and analyses would be useful?
  - How to incorporate system in larger software engineering context