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
Outline

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

Consider two statements

More suspicious of being faulty

\[ m = x \]

\[ w = y \]
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

\[
hue(s) = \text{low hue (red)} + \frac{\%\text{passed}(s)}{\%\text{passed}(s) + \%\text{failed}(s)} \times \text{hue range}
\]
Hue

\[
\text{hue}(s) = \text{low hue (red)} + \frac{\%\text{passed}(s)}{\%\text{passed}(s) + \%\text{failed}(s)} \times \text{hue range}
\]
Brightness

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

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

\[
\frac{10 \text{ failed test cases}}{10} = 100\%
\]

\[
\frac{20 \text{ passed test cases}}{200} = 10\%
\]

\[
m = y;
\]
Example

```c
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;
        else if (x<z)
            6: m = y;
        else
            7: if (x>y)
                8: m = y;
            else if (x>z)
                9: m = x;
            10: m = m;
    11: else if (x>z)
        12: m = x;
    13: print(“Middle number is:”, m);
}
```

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>Pass Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,3,5</td>
<td>P</td>
</tr>
<tr>
<td>1,2,3</td>
<td>P</td>
</tr>
<tr>
<td>3,2,1</td>
<td>P</td>
</tr>
<tr>
<td>5,5,5</td>
<td>P</td>
</tr>
<tr>
<td>5,3,4</td>
<td>P</td>
</tr>
<tr>
<td>2,1,3</td>
<td>F</td>
</tr>
</tbody>
</table>

Pass Status: P P P P P P F
Example

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 = x;
        }
        else {
            if (x> y) {
                m = y;
            } else if (x > z) {
                m = x;
            }
        }
    }
    print(“Middle number is:”, m);
}

Test Cases

<table>
<thead>
<tr>
<th>3,3,5</th>
<th>1,2,3</th>
<th>3,2,1</th>
<th>5,5,5</th>
<th>5,3,4</th>
<th>2,1,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>F</td>
</tr>
</tbody>
</table>

Pass Status: P 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

```c
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
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
Three Faults

Faulty statements

Non-faulty statements
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