Selected Topics of Software Technology 3

Spectrum-based Fault Localization

Birgit Hofer
Institute for Software Technology
We before we start, a few organizational things

- **VO**
  - Date for the written exam?

- **Practical PART 2 – Static Code Analysis**
  - Teams?
Selected Topics of Software Technology 3

Spectrum-based Fault Localization

Visualization

Design & Maintenance Support

Static Analysis

Model-based

Spreadsheet Quality Assurance Techniques

Modeling

Debugging

Fault localization

Testing

Repair

Genetic

Outline – Spectrum-based Fault Localization

- Software
  - Repetition
  - Bank Account Example
  - Influence of the Test suite quality
  - SFL in practice

- Spreadsheets

- Summary
Outline – Spectrum-based Fault Localization

- **Software**
  - Repetition
    - Bank Account Example
    - Influence of the Test suite quality
    - SFL in practice

- **Spreadsheets**

- **Summary**
Relevant Literature

- Abreu, Zoeteweij, Gemund:  
  “An Evaluation of Similarity Coefficients for Software Fault Localization”  

- Lucia, Lo, Jiang, Thung, Budi:  
  “Extended comprehensive study of association measures for fault localization”  

- Hofer, Perez, Abreu, Wotawa:  
  “On the empirical evaluation of similarity coefficients for spreadsheets fault localization”  
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<y)
        7: m = y; // *** bug ***
    8: else
    9: if (x>y)
        10: m = y;
    11: else if (x>z)
        12: m = x;
    13: print("Middle number is: ", m);
}

**Observation Matrix**

<table>
<thead>
<tr>
<th></th>
<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>
<td>F</td>
</tr>
</tbody>
</table>

Error Vector

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

<table>
<thead>
<tr>
<th></th>
<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></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>F</td>
</tr>
</tbody>
</table>

Selected Topics of Software Technology 3

Spectrum - based Fault Localization


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

Pass/Fail Status
```
<table>
<thead>
<tr>
<th></th>
<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>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>F</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
```
```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;
        6: else if (x<z)
            7: m = y; // *** bug ***
    8: else
    9: if (x>y)
10:    m = y;
11: else if (x>z)
12:    m = x;
13: print("Middle number is:", m);
}
```

**Pass/Fail Status**

<table>
<thead>
<tr>
<th></th>
<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>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>P</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>P</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>F</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

```
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; // *** bug ***
    8: else
    9: if (x>y)
        10: m = y;
    11: else if (x>z)
        12: m = x;
    13: print("Middle number is:", m);
}
```

Pass/Fail Status

Spectrum-based Fault Localization (SFL)

- Divide into passed and failed test cases
- Log which statements were executed when executing the single test cases (Observation Matrix)
- Identify those program parts whose execution pattern correlates most with the error vector
```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; // *** bug ***

    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);
}
```

1 match, 5 mismatches

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

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; // *** bug ***
    8:     else
    9:     if (x>y)
   10:         m = y;
   11:     else if (x>z)
   12:         m = x;
   13:     print("Middle number is:", m);
}

Similarity Coefficients

- $a_{11} = $ executed and error
- $a_{10} = $ executed and no error
- $a_{01} = $ not executed and error
- $a_{00} = $ not executed and no error

$\text{Ochiai} = \frac{a_{11}}{\sqrt{(a_{11} + a_{01}) \times (a_{11} + a_{10})}}$

$\text{Jaccard} = \frac{a_{11}}{a_{11} + a_{01} + a_{10}}$

$\text{Stat Bug Isolation} = \frac{a_{11}}{a_{11} + a_{10}}$

$\text{Tarantula} = \frac{a_{11}}{a_{11} + a_{01}} + \frac{a_{10}}{a_{10} + a_{00}}$

There are more than 40 similarity coefficients!
Selected Topics of Software Technology 3
Spectrum-based Fault Localization

Solution for previous Example

```c
mid() {
    int x, y, z, m;
    read("Enter 3 numbers:", x, y, z); // a00 = executed and no error
    m = z; // a01 = not executed and error
    if (y < z) // a11 = executed and error
    {
        if (x < y) // a10 = executed and no error
        {
            m = y; // *** bug ***
        }
    }
    else if (x < z) // a00 = not executed and no error
    {
        m = y;
    }
    else
    {
        if (x > y)
        {
            m = y;
        }
    }
    else if (x > z)
    {
        m = x;
    }
    print("Middle number is:", m); // a00 = executed and no error
}
```

<table>
<thead>
<tr>
<th>a00</th>
<th>a01</th>
<th>a10</th>
<th>a11</th>
<th>Occhaj</th>
<th>Ranking</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>

\[
a_{11} = \frac{a_{11}}{\sqrt{(a_{11} + a_{01}) \times (a_{11} + a_{10})}}
\]
Solution for previous 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;                            
    6: else if (x < z)                   
    7: m = y; // *** bug ***             
    8: else                               
    9: if (x > y)                        
   10: m = y;                           
   11: else if (x > z)                  
   12: m = x;                           
   13: print("Middle number is:", m);  
}
```

<table>
<thead>
<tr>
<th></th>
<th>a₀₀</th>
<th>a₀₁</th>
<th>a₁₀</th>
<th>a₁₁</th>
<th>Occhai</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0,41</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0,41</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0,41</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0,50</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0,00</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0,58</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0,71</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0,00</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0,00</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0,00</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0,00</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0,00</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0,41</td>
<td>4</td>
</tr>
</tbody>
</table>

P P P P P F
Outline – Spectrum-based Fault Localization

- Software
  - Repetition
  - Bank Account Example
  - Influence of the Test suite quality
  - SFL in practice

- Spreadsheets

- Summary
Bank Account Example – Version 1

```java
1. public class BankAccount {
2.     public long balance;
3.     public long limit;
4.     public BankAccount(long bal, long limit) {
5.         this.balance = bal;
6.         this.limit = limit;
7.     }
8.     public void withdraw(long amount) {
9.         if ((balance - amount) >= limit) {
10.            balance = balance - amount;
11.        }
12.    }
13.    public void deposit(long amount) {
14.        balance = balance + amount;
15.    }
16.    public void transferTo(BankAccount acc) {
17.        long money = this.balance;
18.        if (money!=0) { //FAULT
19.            this.withdraw(money);
20.            acc.deposit(money);
```

Bank Account Example – Test Cases (1)

```java
public void testTransfer1 { //T1
    BankAccount a1 = new BankAccount(-100,-1000);
    BankAccount a2 = new BankAccount(0,0);
    a2.deposit(200);
    a1.transferTo(a2);
    Assert.assertEquals(-100, a1.balance);
    Assert.assertEquals(200, a2.balance);
}

public void testWithdraw() { //T2
    BankAccount a1 = new BankAccount(0,-1000);
    a1.withdraw(100);
    Assert.assertEquals(-100, a1.balance);
}

public void testDeposit() { //T3
    BankAccount a2 = new BankAccount(100,0);
    a2.deposit(200);
    Assert.assertEquals(300, a2.balance);
}
Bank Account Example – Test Cases (2)

```java
public void testTransfer2() { //T4
    BankAccount a1 = new BankAccount(0, -1000);
    a1.withdraw(100);
    BankAccount a2 = new BankAccount(0, 0);
    a2.deposit(200);
    a2.transferTo(a1);
    Assert.assertEquals(100, a1.balance);
    Assert.assertEquals(0, a2.balance);
}
public void testTransfer3() { //T5
    BankAccount a1 = new BankAccount(0, -1000);
    BankAccount a2 = new BankAccount(0, 0);
    a2.deposit(200);
    a1.transferTo(a2);
    Assert.assertEquals(0, a1.balance);
    Assert.assertEquals(200, a2.balance);
}
```
## BankAccount Example – Solution

<table>
<thead>
<tr>
<th>Line $s$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>18</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## BankAccount Example – Solution

<table>
<thead>
<tr>
<th>Line $s$</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>Coeff.</th>
<th>Rank.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>0.447</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>0.447</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td>0.577</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>⬤</td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td>0.577</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>0.500</td>
<td>7</td>
</tr>
<tr>
<td>17</td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>0.577</td>
<td>3</td>
</tr>
<tr>
<td><strong>18</strong></td>
<td>⬤</td>
<td></td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>0.577</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.707</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.707</td>
<td>1</td>
</tr>
<tr>
<td><strong>Error</strong></td>
<td>⬤</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation Methods

- **Best Case**
  \[ Rank_{Best} = |\{ c \mid Ochiai(c) > f \}| + 1 \]

- **Average Case**
  \[ Rank_{AVG} = |\{ c \mid Ochiai(c) > f \}| + \frac{|\{ c \mid Ochiai(c) = f \}|}{2} + 0.5 \]

- **Worst Case**
  \[ Rank_{WORST} = |\{ c \mid Ochiai(c) \geq f \}| \]
Outline – Spectrum-based Fault Localization

- Software
  - Repetition
  - Bank Account Example
  - Influence of the Test suite quality
  - SFL in practice

- Spreadsheets

- Summary
The more Test Cases the better the Results
BankAccount Example
What happens when you …

- do not use all test cases?
- duplicate a passing test case?
- duplicate a failing test case?
BankAccount Example
What happens when you …

- do not use all test cases?

<table>
<thead>
<tr>
<th>Line</th>
<th>0 x T2 Occhai</th>
<th>0 x T2 Ranking</th>
<th>0 x T3 Occhai</th>
<th>0 x T3 Ranking</th>
<th>0 x T4 Occhai</th>
<th>0 x T4 Ranking</th>
<th>0 x T5 Occhai</th>
<th>0 x T5 Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.500</td>
<td>7</td>
<td>0.500</td>
<td>8</td>
<td>0.500</td>
<td>8</td>
<td>0.500</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>0.500</td>
<td>7</td>
<td>0.500</td>
<td>8</td>
<td>0.500</td>
<td>8</td>
<td>0.500</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>0.707</td>
<td>1</td>
<td>0.577</td>
<td>3</td>
<td>0.707</td>
<td>3</td>
<td>0.577</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0.707</td>
<td>1</td>
<td>0.577</td>
<td>3</td>
<td>0.707</td>
<td>3</td>
<td>0.577</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>0.500</td>
<td>7</td>
<td>0.577</td>
<td>3</td>
<td>0.577</td>
<td>7</td>
<td>0.577</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>0.577</td>
<td>5</td>
<td>0.577</td>
<td>3</td>
<td>0.707</td>
<td>3</td>
<td>0.707</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>0.577</td>
<td>5</td>
<td>0.577</td>
<td>3</td>
<td>0.707</td>
<td>3</td>
<td>0.707</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>0.707</td>
<td>1</td>
<td>0.707</td>
<td>1</td>
<td>1.000</td>
<td>1</td>
<td>0.707</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>0.707</td>
<td>1</td>
<td>0.707</td>
<td>1</td>
<td>1.000</td>
<td>1</td>
<td>0.707</td>
<td>1</td>
</tr>
</tbody>
</table>
BankAccount Example
What happens when you …

- duplicate a passing test case?
- duplicate a failing test case?

<table>
<thead>
<tr>
<th>Line</th>
<th>2 x T1</th>
<th>2 x T2</th>
<th>2 x T3</th>
<th>2 x T4</th>
<th>2 x T5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Occhai</td>
<td>Ranking</td>
<td>Occhai</td>
<td>Ranking</td>
<td>Occhai</td>
</tr>
<tr>
<td>5</td>
<td>0.447</td>
<td>8</td>
<td>0.408</td>
<td>8</td>
<td>0.408</td>
</tr>
<tr>
<td>6</td>
<td>0.447</td>
<td>8</td>
<td>0.408</td>
<td>8</td>
<td>0.408</td>
</tr>
<tr>
<td>9</td>
<td>0.577</td>
<td>3</td>
<td>0.500</td>
<td>5</td>
<td>0.577</td>
</tr>
<tr>
<td>10</td>
<td>0.577</td>
<td>3</td>
<td>0.500</td>
<td>5</td>
<td>0.577</td>
</tr>
<tr>
<td>14</td>
<td>0.500</td>
<td>7</td>
<td>0.500</td>
<td>5</td>
<td>0.447</td>
</tr>
<tr>
<td>17</td>
<td>0.577</td>
<td>3</td>
<td>0.577</td>
<td>3</td>
<td>0.577</td>
</tr>
<tr>
<td>18</td>
<td>0.577</td>
<td>3</td>
<td>0.577</td>
<td>3</td>
<td>0.577</td>
</tr>
<tr>
<td>19</td>
<td>0.707</td>
<td>1</td>
<td>0.707</td>
<td>1</td>
<td>0.707</td>
</tr>
<tr>
<td>20</td>
<td>0.707</td>
<td>1</td>
<td>0.707</td>
<td>1</td>
<td>0.707</td>
</tr>
</tbody>
</table>
Simple C Compiler

# Statements

# Positive TC

# Negative TC
A Small Case Study

Traffic alert and collision avoidance system

77 NCSS (LOC)
1546 Test Cases (TC)
39 Predef. single faults
Tcas
Fault 38
(wrong array size)

Passing TC: Few
Failing TC: Many
Tcas
Fault 9
(\(\geq\) instead of \(>\))

Passing TC: Not too few
Failing TC: Non-relevant

# Statements

# Positive TC

# Negative TC
Tcas
Fault 30
(missing case)
Tcas
Fault 30
(missing case)

Passing TC: Not to few
Failing TC: Many
Tcas
Fault 35
(missing negation)

Passing TC: Not too few
too many
Failing TC: Many
Tcas

Fault 16
(wrong const. value)

Sometimes it does not matter

# Statements

# Positive TC

# Negative TC
Influence of the Test Suite on the Ranking Result

- Number of failing TC
  - Higher number improves accuracy of the diagnosis
  - Benefit of having more than 10 failing TC is marginal

- Number of passing TC
  - Can have a significant effect (both in positive and negative direction)
  - Effect stabilizes around 20 passing TC

Outline – Spectrum-based Fault Localization

- Software
  - Repetition
  - Bank Account Example
  - Influence of the Test suite quality
  - SFL in practice

- Spreadsheets

- Summary
How to implement?

- Use a code coverage tool
  - Java: EMMA
  - .NET: Ncover

- Memory Optimization
  - Increment $a_{11}$, $a_{10}$, $a_{01}$ instead of storing the whole information matrix
Outline – Spectrum-based Fault Localization

- Software
  - Repetition
  - Bank Account Example
  - Influence of the Test suite quality
  - SFL in practice

- Spreadsheets

- Summary
## Running Example

### Faulty Spreadsheet

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>week 1</td>
<td>week 2</td>
<td>Total</td>
<td>$/h</td>
<td>Gross Pay</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>23</td>
<td>31</td>
<td>23</td>
<td>15</td>
<td>$345,00</td>
</tr>
<tr>
<td>3</td>
<td>Jones</td>
<td>35</td>
<td>34</td>
<td>69</td>
<td>17</td>
<td>$1,173,00</td>
</tr>
<tr>
<td>4</td>
<td>Total</td>
<td>58</td>
<td>65</td>
<td>92</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Formula View

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>week 1</td>
<td>week 2</td>
<td>Total</td>
<td>$/h</td>
<td>Gross Pay</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>23</td>
<td>31 =SUM(B2)</td>
<td>15</td>
<td>=D2*E2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Jones</td>
<td>35</td>
<td>34 =SUM(B3:C3)</td>
<td>17</td>
<td>=D3*E3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Total</td>
<td>=SUM(B2:B3)</td>
<td>=SUM(C2:C3)</td>
<td>=SUM(D2:D3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Test Cases for Spreadsheets

- **Input cells**: cells that do not reference other cells
  \[ I = \{B2=23, C2=31, E2=15, B3=35, C3=34, E3=17\} \]

- **Output cells**: any formula cell, determined by user
  \[ O = \{B4=58, C4=65, D4=123, F2=810, F3=1173\} \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>week 1</td>
<td>week 2</td>
<td>Total</td>
<td>$/h</td>
<td>Gross Pay</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>23</td>
<td>31</td>
<td>23</td>
<td>15</td>
<td>$345,00</td>
</tr>
<tr>
<td>3</td>
<td>Jones</td>
<td>35</td>
<td>34</td>
<td>69</td>
<td>17</td>
<td>$1,173,00</td>
</tr>
<tr>
<td>4</td>
<td>Total</td>
<td>58</td>
<td>65</td>
<td>92</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
From 3rd level programs to spreadsheets

Program debugging: execution traces, slices
Spreadsheets: cones (borrowed from hardware debugging)

\[ \text{CONE}(c) = c \cup \bigcup_{c' \in \rho(c)} \text{CONE}(c') \]

The function \( \rho(c) \) returns all cells referenced in \( c \).
Example for cones

- $\text{CONE}(F2) = \{B2, D2, E2, F2\}$
- $\text{CONE}(D4) = \{B2, D2, B3, C3, D3, D4\}$

Investigating Intersection of cones
Faults where $\cap$ of cones does not work

- Several faults

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>week 1</td>
<td>week 2</td>
<td>Total =SUM(B2:C2)</td>
<td>$/h$</td>
<td>Gross Pay=D2*C2</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>23</td>
<td>31</td>
<td>=SUM(B3:C3)</td>
<td>15</td>
<td>=D3*C3</td>
</tr>
<tr>
<td>3</td>
<td>Jones</td>
<td>35</td>
<td>34</td>
<td>=SUM(D2:D3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Total</td>
<td>=SUM(B2:B3)</td>
<td>=SUM(C2:C3)</td>
<td>=SUM(D3:D3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Cone(F2) = \{B2, C2, D2, F2\}
- Cone(F3) = \{B3, C3, D3, F3\}

- Single wrong output cell

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>week 1</td>
<td>week 2</td>
<td>Total =SUM(B2:C2)</td>
<td>$/h$</td>
<td>Gross Pay=D2*E2</td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>23</td>
<td>31</td>
<td>=SUM(B3:C3)</td>
<td>15</td>
<td>=D3*E3</td>
</tr>
<tr>
<td>3</td>
<td>Jones</td>
<td>35</td>
<td>34</td>
<td>=SUM(D3:D3)</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Total</td>
<td>=SUM(B2:B3)</td>
<td>=SUM(C2:C3)</td>
<td>=SUM(D3:D3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Cone(F3) = \{B3, C3, D3, D4, F3\}
SFL for spreadsheets

- $a_{11} = \text{in cone} \& \text{error}$
- $a_{10} = \text{in cone} \& \text{no error}$
- $a_{01} = \text{not in cone} \& \text{error}$
- $a_{00} = \text{not in cone} \& \text{no error}$

\[
\begin{align*}
\text{Ochiai} &= \frac{a_{11}}{\sqrt{(a_{11} + a_{01}) \times (a_{11} + a_{10})}} \\
\text{Jaccard} &= \frac{a_{11}}{a_{11} + a_{01} + a_{10}} \\
\text{Stat Bug Isolation} &= \frac{a_{11}}{a_{11} + a_{10}} \\
\text{Tarantula} &= \frac{a_{11}}{a_{11} + a_{01}} + \frac{a_{10}}{a_{10} + a_{00}}
\end{align*}
\]
Spectrum-based Fault Localization

- **Spectra:**
  Cones of faulty and correct output cells

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>week 1</td>
<td>week 2</td>
<td>Total</td>
<td>$/h</td>
<td>Gross Pay</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>23</td>
<td>31</td>
<td>=SUM(B2)</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Jones</td>
<td>35</td>
<td>34</td>
<td>=SUM(B3:C3)</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Total</td>
<td>=SUM(B2:B3)</td>
<td>=SUM(C2:C3)</td>
<td>=SUM(D2:D3)</td>
<td></td>
</tr>
</tbody>
</table>

CONE(F2) = {B2,D2,E2,F2}
CONE(D4) = {B2,D2,B3,C3,D3,D4}
CONE(B4) = {B2,B3,B4}
CONE(C4) = {C2,C3,C4}
CONE(F3) = {B3,C3,D3,E3,F3}
Spectrum-based Fault Localization

- **Spectra:**
  - Cones of faulty and correct output cells

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>week 1</td>
<td>week 2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Jones</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>Total</td>
<td>=SUM(B2:B3)</td>
</tr>
</tbody>
</table>

CONE(F2) = {B2,D2,E2,F2}
CONE(D4) = {B2,D2,B3,C3,D3,D4}
CONE(B4) = {B2,B3,B4}
CONE(C4) = {C2,C3,C4}
CONE(F3) = {B3,C3,D3,E3,F3}

<table>
<thead>
<tr>
<th></th>
<th>F2</th>
<th>D4</th>
<th>B4</th>
<th>C4</th>
<th>F3</th>
<th>Coef.</th>
<th>Rank.</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Spectrum-based Fault Localization

### Spectra:

Cones of faulty and correct output cells

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Green</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Jones</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td><strong>Total</strong></td>
<td><strong>=SUM(B2:B3)</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cone (Cells)</th>
<th>Coef.</th>
<th>Rank.</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>0.816</td>
<td>2</td>
</tr>
<tr>
<td>D4</td>
<td>0.707</td>
<td>3</td>
</tr>
<tr>
<td>B4</td>
<td>0.707</td>
<td>3</td>
</tr>
<tr>
<td>C4</td>
<td>0.707</td>
<td>3</td>
</tr>
<tr>
<td>F3</td>
<td>0.707</td>
<td>3</td>
</tr>
<tr>
<td>Error</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CONE(F2) = {B2,D2,E2,F2}**

**CONE(D4) = {B2,D2,B3,C3,D3,D4}**

**CONE(B4) = {B2,B3,B4}**

**CONE(C4) = {C2,C3,C4}**

**CONE(F3) = {B3,C3,D3,E3,F3}**
Demo
Influence of the “Test suite” — Avg. Rank

No user wants to indicate for so many output cells if they are correct.

Research questions

RQ1: Do spreadsheets contain correct output cells that positively or negatively influence the ranking of the faulty cells?

RQ2: If yes, is it possible to a-priori determine which correct output cells would positively influence the ranking?

RQ3: Is it possible to avoid a decreasing fault localization quality when adding more correct output cells?
RQ1: Do spreadsheets contain correct output cells that positively or negatively influence the ranking of the faulty cells?

EUSES my_financial_model
**RQ1:** Do spreadsheets contain correct output cells that positively or negatively influence the ranking of the faulty cells?

**ISCAS85 c7552**

![Box plot diagram showing the relationship between the number of correct output cells and the fault ranking.](image)
RQ2: If yes, is it possible to a-priori determine which correct output cells would positively influence the ranking?

- Avoid coincidental correct output cells
  - A-priori definition not possible
  - Too many potential coincidental correct output cells

- Take output cells with largest cones first

<table>
<thead>
<tr>
<th>Rank_{AVG} for one correct output cell</th>
<th>Random selection</th>
<th>Largest cone</th>
</tr>
</thead>
</table>
| EUSES  
my_financial                     | 5.8             | 2.5          |
| ISCAS85  
c7552                          | 100.6           | 69.5         |
Coincidental Correctness

- Conditional like IF-function
- Abstraction function like MIN, MAX, COUNT
- Boolean
- Multiplication by zero
- Power with 0 or 1 as base number or 0 as exponent

Coincidental Correctness is also a problem for other software (see mid-Example)
**RQ3:** Is it possible to avoid a decreasing fault localization quality when adding more correct output cells?

- Balance ratio of correct and erroneous output cells
- Duplicate cones of erroneous output cells

**ISCAS85 c7552**

[Diagrams showing the relationship between correct and erroneous output cells and their impact on rank average.]
Research questions - Summary

RQ1

Do spreadsheets contain correct output cells that positively or negatively influence the ranking of the faulty cells?

YES

RQ2

If yes, is it possible to a-priori determine which correct output cells would positively influence the ranking?

YES (use largest cones first)

RQ3

Is it possible to avoid a decreasing fault localization quality when adding more correct output cells?

YES (duplicate cones of erroneous output cells)
Outline – Spectrum-based Fault Localization

- Software
  - Repetition
  - Bank Account Example
  - Influence of the Test suite quality
  - SFL in practice

- Spreadsheets

- Summary
## Summary

<table>
<thead>
<tr>
<th>Approach</th>
<th>SFL</th>
</tr>
</thead>
<tbody>
<tr>
<td>User input</td>
<td>correct / faulty</td>
</tr>
<tr>
<td>Fault Complexity</td>
<td>single faults</td>
</tr>
<tr>
<td>Computational Complexity</td>
<td>low</td>
</tr>
<tr>
<td>Granularity</td>
<td>block level</td>
</tr>
<tr>
<td>Result</td>
<td>ranking</td>
</tr>
</tbody>
</table>