CUES FOR SCENT
INTENSIFICATION IN DEBUGGING

Alexandre Perez and Rui Abreu
Faculty of Engineering
University of Porto, Portugal

November 4th 2013,
International Workshop on Program Debugging
Fault Localization

```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;
        // ***BUG***
    } else {
        if (x>y) m = y;
        else if (x>z) m = x;
    }
    print("Middle number is:",m);
}
```

<table>
<thead>
<tr>
<th>Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>Suspiciousness</td>
</tr>
<tr>
<td>0.41</td>
</tr>
<tr>
<td>0.41</td>
</tr>
<tr>
<td>0.41</td>
</tr>
<tr>
<td>0.50</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.58</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
</tr>
</tbody>
</table>

0.41 0.41 0.41 0.50 0.0 0.58 0.71

Runs: 1 2 3 4 5 6
Suspiciousness: 0.41 0.41 0.41 0.50 0.0 0.58 0.71
FAULT LOCALIZATION

Are Automated Debugging Techniques Actually Helping Programmers?

Chris Parnin and Alessandro Orso
Georgia Institute of Technology
College of Computing
{chris.parnin|orso}@gatech.edu
**Information Foraging**

**Predator:** person debugging the program.

**Prey:** what the predator seeks to know to pinpoint the fault.
**Information Foraging**

**Proximal cues:** runtime behaviors that suggest information relative to the prey.

**Information scent:** perceived likelihood that a cue will potentially lead to the fault.

**Topology:** is the collection of paths through which the programmer can navigate.
**INFORMATION FORAGING**

\[
\text{choice} = \max \left( \frac{\text{information gain}}{\text{cost}} \right)
\]
Sunburst
SUNBURST
SUNBURST

① Project


**Sunburst**

① Project

② Source Folder
Sunburst

① Project
② Source Folder
③ Package
SUNBURST

① Project
② Source Folder
③ Package
④ File
SUNBURST

1. Project
2. Source Folder
3. Package
4. File
5. Class
SUNBURST

1. Project
2. Source Folder
3. Package
4. File
5. Class
6. Method
SUNBURST

1. Project
2. Source Folder
3. Package
4. File
5. Class
6. Method
7. Statement
Sunburst

High Suspiciousness

Low Suspiciousness
SUNBURST
SUNBURST

Root Change
Check it out at www.gzoltar.com
**INFORMATION FORAGING**

\[
\text{choice} = \max \left( \frac{\text{information gain}}{\text{cost}} \right)
\]
**INFORMATION FORAGING**

\[ \text{choice} = \max \left( \frac{\text{information gain}}{\text{cost}} \right) \]

- Color coding and the hierarchical visualization increase information scent.
- Interaction features can reduce the cost of navigation.
USER STUDY

40 Participants
User Study

Students of the Master in Informatics and Computing Engineering at University of Porto
User Study

5+ years of Java/JUnit experience
(De)serializes Java objects into XML

17 kLOC, 306 classes

Participants were unfamiliar with the application

http://xstream.codehaus.org/
final List<Object> parameter = new ArrayList<Object>();
if (targetType == null) {
    parameter.add(targetType);
}
final List<Object> arrays = new ArrayList<Object>();
final List<Object> parameter = new ArrayList<Object>();
if (targetType == null) {
    parameter.add(targetType);
}
final List<Object> arrays = new ArrayList<Object>();

== instead of !=
Participants were asked to locate and fix the fault.

**Metric:** time-to-green.

**Timeout:** 30 minutes.
CONTROL GROUP

Find and fix the fault without the aid of GZoltar.

7 out of 20 participants completed the task.
**CONTROL GROUP**

![Diagram showing two rows of people, with the top row in green and the bottom row in red.]

**Average time:** 23.4 minutes*

*Those that did not find the fault were assigned as taking the maximum time (30 minutes)*
EXPERIMENTAL GROUP

Find and fix the fault with the aid of GZoltar.

All participants completed the task.
EXPERIMENTAL GROUP

Average time: 7.9 minutes
Fault Localization

```c
void 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;
        // ***BUG***
    } else {
        if (x>y) m = y;
        else if (x>z) m = x;
    }
    m = y; // ***BUG***
} else {
    if (x>y) m = y;
    else if (x>z) m = x;
}
print("Middle number is:", m);
```
Fault Localization

```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;
        // ***BUG***
    } else {
        if (x > y)
            m = y;
        else if (x > z)
            m = x;
    }
    else if (x < z)
        m = y;
    }
    else {
        if (x > z)
            m = x;
    }
    print("Middle number is:", m);
}
```

<table>
<thead>
<tr>
<th>Runs</th>
<th>Suspiciousness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.41</td>
</tr>
<tr>
<td>2</td>
<td>0.41</td>
</tr>
<tr>
<td>3</td>
<td>0.50</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>0.58</td>
</tr>
<tr>
<td>6</td>
<td>0.71</td>
</tr>
</tbody>
</table>

**Information Foraging**

**Proximal cues:** runtime behaviors that suggest information relative to the prey.

**Information scent:** perceived likelihood that a cue will potentially lead to the fault.

**Topology:** is the collection of paths through which the programmer can navigate.
**Fault Localization**

```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);
}
```

**Information Foraging**

**Proximal cues:** runtime behaviors that suggest information relative to the prey.

**Information scent:** perceived likelihood that a cue will potentially lead to the fault.

**Topology:** is the collection of paths through which the programmer can navigate.

**Sunburst**

1. Project
2. Source Folder
3. Package
4. File
5. Class
6. Method
7. Statement
**Fault Localization**

```java
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);
}
```

**Information Foraging**

**Proximal cues:** runtime behaviors that suggest information relative to the prey.

**Information scent:** perceived likelihood that a cue will potentially lead to the fault.

**Topology:** is the collection of paths through which the programmer can navigate.

---

**Sunburst**

1. Project
2. Source Folder
3. Package
4. File
5. Class
6. Method
7. Statement

---

**Graph**

- **With Visualizations**
- **Without Visualizations**

**Y-axis:** Density
**X-axis:** Seconds