Fault Seeding vs. Mutation Operators: An Empirical Comparison of Testing Techniques for Web Applications

1. General Research Problem
   - Experimenters need faults to evaluate testing techniques
   - Three known methods for obtaining faults

<table>
<thead>
<tr>
<th>Technique</th>
<th>Advantages</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>manually seeding faults</td>
<td>based on developer's knowledge</td>
<td>• time consuming</td>
</tr>
<tr>
<td></td>
<td>of application and its</td>
<td>• faults may not be realistic</td>
</tr>
<tr>
<td></td>
<td>implementation</td>
<td></td>
</tr>
<tr>
<td>using mutation operators</td>
<td>quickly generates many</td>
<td>• equivalent mutants possible</td>
</tr>
<tr>
<td></td>
<td>faulty versions</td>
<td>• faults may not be realistic</td>
</tr>
<tr>
<td>obtaining real faulty</td>
<td>are real faults</td>
<td>• difficult to obtain</td>
</tr>
<tr>
<td>versions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. MuJava: Mutation System for Java
   - Apply mutation operators, provides a more realistic sample of faults?

5. Methodology for Generating and Detecting Faults
   - Mutants Generator
     - original program (source code)
     - compile
     - original program (byte code)
     - Mutants Generator
       - behavioral mutants generator
       - structural mutants generator
     - behavioral mutants
     - structural mutants

6. Results
   - E1. Manual-Seeded
     - Total Faults: 30
     - Faults Detected: 18
     - Detection Rate: 60%
   - E2. Mutants
     - Total Faults: 341
     - Faults Detected: 288
     - Detection Rate: 84.50%

7. Analysis
   - Faults caught by 85%/169 test cases are probably unrealistic
   - Faults caught by 1 to 10 test cases (0 to 10 for manual seeded) are probably the most realistic
   - Not yet empirically proven with real faults
   - 56.5% of manual seeded faults fall in the realistic range, compared to 41.9% of mutants
   - A1. With more investigation, this result could show that manual seeding creates more realistic faults than creating mutants.
   - A2. MuJava does not provide sufficient mutation operators for web application testing (e.g., form, link, and appearance faults)

8. Contributions
   - Applied MuJava’s mutation operator to web applications
   - Empirical comparison of mutants for web applications
   - Identified limits of operators for web applications

9. Future Work
   - Evaluate approach using other subject web applications
   - Compare results with results from real faults
   - Apply MuJava to Java code generated from JSPs
   - Create web-application-specific mutation operators

Funded by the CRA-W Distributed Mentor Program