Using Run Time Traces in Automated Programming Tutoring

Michael Striewe, Michael Goedicke
University of Duisburg-Essen
Motivation

• Tests are a typical and common way to check programming assignments
• Students have to relate test results to program behaviour
  • Requires thinking in program states, steps, ...
  • Can be supported by detailed information about variable values
• Helpful information provided by debugging tools, but:
  • Additional cognitive load for novice programmers (concepts + language + tools)
  • Wrong use of tools forces wrong interpretation of program behaviour
• Alternative approach:
  • Provide traces automatically
  • Offer automated trace analysis
Example

```java
public float minDiagonal(int[][] mat){
    float meanValue = 0.0f;
    float minMain=0, minSecond=0;
    for(int i=0; i<mat.length-1; i++){
        for(int j=0; j<mat.length-1; j++){
            if(mat[i][j]>mat[i+1][j+1]){
                minMain=mat[i+1][j+1];
            }
        }
    }
    meanValue=minMain+1;
    return meanValue;
}
```

<table>
<thead>
<tr>
<th>File and line</th>
<th>Variable values</th>
<th>minMain</th>
<th>minSecond</th>
<th>meanValue</th>
<th>Line of code to be executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testat3:42</td>
<td>{10,2,1},{4,3,2},{2,2,7}</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>float meanValue = 0.0f;</td>
</tr>
<tr>
<td>Testat3:43</td>
<td>{10,2,1},{4,3,2},{2,2,7}</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>float minMain=0, minSecond=0;</td>
</tr>
<tr>
<td>Testat3:44</td>
<td>{10,2,1},{4,3,2},{2,2,7}</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>for(int i=0; i&lt;mat.length-1; i++){</td>
</tr>
<tr>
<td>Testat3:45</td>
<td>0 {10,2,1},{4,3,2},{2,2,7}</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>for(int j=0; j&lt;mat.length-1; j++){</td>
</tr>
<tr>
<td>Testat3:46</td>
<td>0 0 {10,2,1},{4,3,2},{2,2,7}</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>if(mat[i][j]&gt;mat[i+1][j+1]){</td>
</tr>
<tr>
<td>Testat3:47</td>
<td>0 0 {10,2,1},{4,3,2},{2,2,7}</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>minMain=mat[i+1][j+1];</td>
</tr>
<tr>
<td>Testat3:48</td>
<td>0 0 {10,2,1},{4,3,2},{2,2,7}</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>for(int j=0; j&lt;mat.length-1; j++){</td>
</tr>
<tr>
<td>Testat3:49</td>
<td>0 1 {10,2,1},{4,3,2},{2,2,7}</td>
<td>3.0</td>
<td>0.0</td>
<td>0.0</td>
<td>if(mat[i][j]&gt;mat[i+1][j+1]){</td>
</tr>
<tr>
<td>Testat3:50</td>
<td>0 1 {10,2,1},{4,3,2},{2,2,7}</td>
<td>3.0</td>
<td>0.0</td>
<td>0.0</td>
<td>for(int j=0; j&lt;mat.length-1; j++){</td>
</tr>
<tr>
<td>Testat3:51</td>
<td>0 1 {10,2,1},{4,3,2},{2,2,7}</td>
<td>3.0</td>
<td>0.0</td>
<td>0.0</td>
<td>if(mat[i][j]&gt;mat[i+1][j+1]){</td>
</tr>
<tr>
<td>Testat3:52</td>
<td>1 {10,2,1},{4,3,2},{2,2,7}</td>
<td>3.0</td>
<td>0.0</td>
<td>0.0</td>
<td>for(int j=0; j&lt;mat.length-1; j++){</td>
</tr>
<tr>
<td>Testat3:53</td>
<td>1 0 {10,2,1},{4,3,2},{2,2,7}</td>
<td>3.0</td>
<td>0.0</td>
<td>0.0</td>
<td>if(mat[i][j]&gt;mat[i+1][j+1]){</td>
</tr>
<tr>
<td>Testat3:54</td>
<td>1 0 {10,2,1},{4,3,2},{2,2,7}</td>
<td>3.0</td>
<td>0.0</td>
<td>0.0</td>
<td>minMain=mat[i+1][j+1];</td>
</tr>
<tr>
<td>Testat3:55</td>
<td>1 0 {10,2,1},{4,3,2},{2,2,7}</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>for(int j=0; j&lt;mat.length-1; j++){</td>
</tr>
<tr>
<td>Testat3:56</td>
<td>1 1 {10,2,1},{4,3,2},{2,2,7}</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>if(mat[i][j]&gt;mat[i+1][j+1]){</td>
</tr>
<tr>
<td>Testat3:57</td>
<td>1 1 {10,2,1},{4,3,2},{2,2,7}</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>for(int j=0; j&lt;mat.length-1; j++){</td>
</tr>
<tr>
<td>Testat3:58</td>
<td>1 {10,2,1},{4,3,2},{2,2,7}</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>for(int i=0; i&lt;mat.length-1; i++){</td>
</tr>
<tr>
<td>Testat3:59</td>
<td>{10,2,1},{4,3,2},{2,2,7}</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>meanValue=minMain+1;</td>
</tr>
<tr>
<td>Testat3:60</td>
<td>{10,2,1},{4,3,2},{2,2,7}</td>
<td>2.0</td>
<td>0.0</td>
<td>3.0</td>
<td>return meanValue;</td>
</tr>
</tbody>
</table>
Technical Remarks

- Two JVMs communicating via JDI
- Test driver executed by tracing framework
- Remote control for aborting test cases
- Arbitrary generation of messages by the test driver and/or tracing framework
- Tracing slows down execution by factor 100!
  - Truncation of traces necessary to get fast feedback
Use Cases (1): Manual Inspection

• Reading traces partly or from beginning to end
• Trying to understand program behaviour
• Trying to get input parameters
• Results:
  • Considered helpful
  • Students got interested in debugging techniques
• Problems:
  • Limited length of traces may cut off important parts
  • Layout / Visibility of large traces on small screens
• Optimizations:
  • Explicit statements for switching tracing off and on via the test driver
Use Cases (2): Assertion Checking

- On-the-fly check for values of private fields or local variable values
- Post-mortem analysis of traces for complex constraints
- Results:
  - On-the-fly check works as expected if anchors are known
- Problems:
  - Hard to identify really good practical scenarios
  - Pre-processing needed to identify relevant variables
  - Analysis imprecise if trace is truncated
Use Cases (3): Trace Alignment

• Comparison of traces from students’ solutions with traces from sample solution
• Using string alignment algorithms
• Expected results:
  • Matching sections interrupted by non-matching sections: Solution is partly correct, but contains wrong or useless code
  • No match from on point on: Solution is correct up to this point
• Implementation:
  • To be done...
Future Work

• Technical:
  • Trace alignment and more automated trace analysis
  • Trace layout
• Research:
  • Studies on program comprehension among students
  • Coupling of traces with other feedback
• Answering your questions!