Data-flow and Control-flow Criteria Compared

- Coverage criteria monitor the thoroughness of software tests
  - Control-flow based
  - Data-flow based
- Are they effective?
- Which ones are more effective?

Experiments

- Goals
  - Comparing effectiveness of data-flow coverage and control-flow coverage for fault-detection
  - Is it necessary to achieve 100% coverage to benefit from a criterion?
- Criteria
  - Data-flow
  - Edge coverage
    - Extends branch coverage by considering both explicit and implicit control-flow in Boolean expressions
    - \( \text{IF} (a \land b \land c) \text{ THEN } x=5 \text{ ELSE } x=10 \) has 6 edges, not 2

Model of Coverage-based Testing

Base Programs

- 7 moderate sized programs

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Executable Edges</th>
<th>DU</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace</td>
<td>512</td>
<td>191</td>
<td>664</td>
<td>pattern replace</td>
</tr>
<tr>
<td>tca5</td>
<td>141</td>
<td>46</td>
<td>57</td>
<td>altitude separation</td>
</tr>
<tr>
<td>usl.123</td>
<td>472</td>
<td>97</td>
<td>268</td>
<td>lexical analyzer</td>
</tr>
<tr>
<td>usl.128</td>
<td>399</td>
<td>159</td>
<td>240</td>
<td>lexical analyzer</td>
</tr>
<tr>
<td>schedule1</td>
<td>292</td>
<td>62</td>
<td>294</td>
<td>priority scheduler</td>
</tr>
<tr>
<td>schedule2</td>
<td>301</td>
<td>80</td>
<td>217</td>
<td>priority scheduler</td>
</tr>
<tr>
<td>tot_info</td>
<td>440</td>
<td>83</td>
<td>292</td>
<td>information measure</td>
</tr>
</tbody>
</table>

How Do We Proceed?

- Generate test cases according to criteria
  - How many test cases?
    - Say we decide on a number \( N \)
  - What coverage?
    - Say 100%
- Execute them on the programs
  - How to detect faults?
  - What if no faults are found?
- Discussion

Fault Space

- Seed faults in the programs
- Ideal world
  - Real faults that have been recorded in the course of development of production software
- Real world
  - Seeded "realistic" faults
    - Mostly changes to single line of code
    - Simple mutations or missing code
    - Sometimes multiple changes
  - Requirements on seeded faults
    - Neither too easy nor too difficult to detect
Fault Space

- **Why?**
  - If too easy then all tests would detect them, irrespective of the coverage
  - If too difficult, then none would detect - no difference in techniques
- **Objective measure of “reasonable” fault**
  - Too difficult if less than LB test cases detect it
  - Too easy if more than UB test cases detect
- **10 people seeded faults**
  - LB = 3; UB = 350
  - 55 were too difficult, 113 were too easy
  - 130 were reasonable; were included in study

Test Oracle

- The original program was assumed to be “correct” and used as an Oracle

Now How Do We Proceed?

- Generate test cases
- Execute them on the programs/mutants
- Record the faults detected

- Any problems with test case generation?
  - Do two test suites that satisfy a coverage criterion have the same fault detection ability?
- Discussion

Test Pool

- Use 2-3 testers to create a test pool
- Randomly select test cases from this test pool

Creation of Test Pool

- Realistic process
- **Create initial test pool (ITP)**
  - Category-partition method
- Examine coverage; identify missing areas
- **Create additional test pool (ATP)**
  - Goal
    - Each exercisable coverage unit is covered by at least 30 test cases
  - Run each test case in the pool and record the outcome (fault detected vs. undetected) and the list of edges and DUs exercised

Test Pool Data

<table>
<thead>
<tr>
<th>Base Program</th>
<th>Number of faulty versions</th>
<th>Test Pool (TP) Initial Tests (ITP)</th>
<th>Additional Tests (ATP)</th>
<th>Final Size Tests (ITP + ATP)</th>
<th>Range of failure ratios in the test pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace</td>
<td>32</td>
<td>79%</td>
<td>21%</td>
<td>5548</td>
<td>.0025-0.056</td>
</tr>
<tr>
<td>war</td>
<td>30</td>
<td>65%</td>
<td>35%</td>
<td>1562</td>
<td>.0006-0.084</td>
</tr>
<tr>
<td>usl.123</td>
<td>7</td>
<td>99%</td>
<td>1%</td>
<td>4092</td>
<td>.0007-0.056</td>
</tr>
<tr>
<td>usl.128</td>
<td>10</td>
<td>99%</td>
<td>1%</td>
<td>4076</td>
<td>.0079-0.086</td>
</tr>
<tr>
<td>schedule1</td>
<td>9</td>
<td>90%</td>
<td>10%</td>
<td>2637</td>
<td>.0027-1.00</td>
</tr>
<tr>
<td>schedule2</td>
<td>10</td>
<td>77%</td>
<td>23%</td>
<td>2666</td>
<td>.0008-0.024</td>
</tr>
<tr>
<td>net_info</td>
<td>23</td>
<td>64%</td>
<td>36%</td>
<td>1067</td>
<td>.0019-1.59</td>
</tr>
</tbody>
</table>

- **usl.128**
  - Test pool size = 4076 cases
  - Hardest fault detected by 32 cases
  - Easiest detected by 350 cases
Generating Test Sets

- **Goal**
  - 5000 test sets for each faulty program
- **For each test set of size N**
  - Randomly take a test case from pool
  - If it increases coverage, add it
  - Until N tests or 100% coverage
- **Sizes**
  - Chosen randomly from 1, 2, ..., R, where R was determined for each program by trial-and-error as the number slightly larger than the size of the largest test set reaching 100% coverage
  - At least 30 tests for each 2% coverage interval

Observations

- In general, the performance of both coverage varied widely
- Program classification
  - According to the method that seemed most effective in detecting its faults
  - Define relations
    - DU > Edge
    - Edge > DU
    - DU > Random
    - Edge > Random
    - Random > DU
    - Random > Edge

Better Analysis

- For each faulty program
  - Fit second order, least squares curves
    - Coverage (F_{DU}, F_{Edge})
    - and size plots (F_{SDU}, F_{SEdge})
- **Definition**
  - DU > Edge if
    - F_{DU}(100%) > F_{Edge}(100%)
    - And (F_{DU}(100%) - F_{Edge}(100%)) > (standard deviation of the difference between the measured fault detection ratio and their least squares approximation)
- F_{random}(s)
  - Given a test set size s
  - Probability that a randomly chosen set of s test cases from the test pool contains at least one fault-detecting test case
  - Expected fault-detection ratio of random test sets of size s
  - Always computed from TP or ITP
  - Avoids bias in favor of coverage
Better Analysis

- For DU coverage
  - Largest test set generated = d
  - Maximum value of $F_{SDU(s)}$ for $s = 1...d = Max_{DU}$

- Similarly, For edge coverage
  - Largest test set generated = e
  - Maximum value of $F_{SEdge(s)}$ for $s = 1...e = Max_{Edge}$

- Definitions
  - DU > Random if $Max_{DU} > Frandom(d)$
  - Edge > Random if $Max_{Edge} > Frandom(e)$
  - And differences satisfy a similar property for DU > Edge and Edge > DU

- Similarly, DU < Random if $Max_{DU} < Frandom(d)$
  and Edge < Random if $Max_{Edge} < Frandom(e)$

Classification of Faults

<table>
<thead>
<tr>
<th>Class</th>
<th>Characteristics</th>
<th>Number of faults</th>
<th>Fault Detection Ratio at 100% coverage min, avg, max</th>
</tr>
</thead>
<tbody>
<tr>
<td>DU</td>
<td>DU &gt; Edge and DU &gt; Random</td>
<td>5</td>
<td>.19, .67, 1.0</td>
</tr>
<tr>
<td>Edge</td>
<td>DU &gt; DU and Edge &gt; Random</td>
<td>12</td>
<td>.27, .57, .99</td>
</tr>
<tr>
<td>DU-Edge</td>
<td>DU &gt; Random and Edge &gt; Random and not (DU &gt; Edge or Edge &gt; DU)</td>
<td>13</td>
<td>.14, .59, 1.0</td>
</tr>
<tr>
<td>Coverage Total</td>
<td>DU &gt; Random or Edge &gt; Random</td>
<td>63</td>
<td>-</td>
</tr>
<tr>
<td>Non-Coverage</td>
<td>DU &lt; Random and Edge &lt; Random</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Others</td>
<td>cannot classify</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

Detection ratios were very low (4)

DU coverage vs. Random

<table>
<thead>
<tr>
<th>% DU Coverage</th>
<th>95-95%</th>
<th>95-99%</th>
<th>99-97%</th>
<th>97-99%</th>
<th>99-100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>average size of DU coverage test sets</td>
<td>7.8</td>
<td>8.3</td>
<td>11.3</td>
<td>14.2</td>
<td>17.4</td>
</tr>
<tr>
<td>average fault detection ratio of DU coverage test sets</td>
<td>.30</td>
<td>.25</td>
<td>.33</td>
<td>.42</td>
<td>.51</td>
</tr>
<tr>
<td>average % superiority in fault detection of DU coverage test sets over same size random test sets</td>
<td>1%</td>
<td>14%</td>
<td>33%</td>
<td>52%</td>
<td>68%</td>
</tr>
<tr>
<td>average % increase in the size of random test sets required to yield the same fault detection as the DU coverage test sets</td>
<td>*</td>
<td>21%</td>
<td>49%</td>
<td>79%</td>
<td>149%</td>
</tr>
</tbody>
</table>

* The observed difference is not statistically significant (less than 95% confidence).

Edge Coverage vs. Random

<table>
<thead>
<tr>
<th>% Edge Coverage</th>
<th>95-95%</th>
<th>95-99%</th>
<th>99-97%</th>
<th>97-99%</th>
<th>99-100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>average size of Edge coverage test sets</td>
<td>7.5</td>
<td>8.5</td>
<td>11.7</td>
<td>14.2</td>
<td>17.6</td>
</tr>
<tr>
<td>average fault detection ratio of Edge coverage test sets</td>
<td>.28</td>
<td>.31</td>
<td>.35</td>
<td>.41</td>
<td>.45</td>
</tr>
<tr>
<td>average % superiority in fault detection of Edge coverage test sets over same size random test sets</td>
<td>40%</td>
<td>48%</td>
<td>50%</td>
<td>68%</td>
<td>75%</td>
</tr>
<tr>
<td>average % increase in the size of random test sets required to yield the same fault detection as the Edge coverage test sets</td>
<td>51%</td>
<td>64%</td>
<td>77%</td>
<td>112%</td>
<td>161%</td>
</tr>
</tbody>
</table>

DU Coverage vs. Edge Coverage

<table>
<thead>
<tr>
<th>% Coverage</th>
<th>95-97%</th>
<th>97-99%</th>
<th>99-100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>average % difference in size of DU coverage test sets over Edge coverage test sets</td>
<td>1%</td>
<td>9%</td>
<td>21%</td>
</tr>
<tr>
<td>average % difference in fault detection of DU coverage test sets over Edge coverage test sets</td>
<td>*</td>
<td>*</td>
<td>38%</td>
</tr>
</tbody>
</table>

* The observed difference is not statistically significant (less than 95% confidence).