Test Prioritization

17-654/17-765
Analysis of Software Artifacts
Jonathan Aldrich

Test Prioritization: Motivation

- Goal: find and eliminate newly introduced defects
- Regression Testing for Windows
  - Many tests
  - Many platform configurations to run them on
  - Full tests take weeks to run
- Test Prioritization
  - Want to run tests likely to fail first
  - Day 1 after internal release, not day 21!
- Test Selection
  - What tests should I run before checking in code?
  - What tests should be run before releasing a critical fix?
  - Special case of prioritization

Observation: New defects are introduced from changed code

Challenges in Test Prioritization

- Detecting change and affected parts of the program
- Scalability to handle complex systems
  - Tens of millions of tests
  - Thousands of developers and testers
  - Tens of millions lines of source code
  - Acceptable response times
- Integrating seamlessly into development process

Scout: Test Prioritization System

- Detect impacted blocks
- Detect impacted blocks likely to be covered by existing tests
- What order should tests be run?
- Detect minimal set of test cases likely to cover the impacted blocks

BMAT – Binary Matching

- Goal: detect corresponding blocks in old and new versions of a program
  - [Wang, Pierce, and McFarling JILP 2000]
- Matches basic blocks in binary code
  - don’t need source code
  - must ignore changes in address space
- Algorithm considers similarities in code and in its uses

BMAT – Matching Procedures

- Match procedures if names match
  - Qualified by package, scope, etc.
  - If ambiguous, extend to include argument types
- Check for similar names
  - Verify match if blocks are similar (see below)
- Look for function bodies hashing the same
- Pairwise compare blocks otherwise
- If no match, conclude function is new
BMAT – Matching Blocks

- Match blocks based on hash of contents
  - Look for exact match first, then apply fuzzy hashing algorithm
  - Fuzzy algorithms ignore information that is likely to change due to innocuous changes: offsets, registers, block adds, opcodes
- Control-flow match
  - Build CFG, look for node pairs with the same connectivity
  - May match many new blocks to one old block
  - Partial match: new block not always executed (e.g. b2’’)

Detecting Impacted Blocks

- Old blocks
  - Identical (modulo address changes)
- Impacted blocks
  - Old modified blocks
  - New blocks

Scout: Test Prioritization System

- What changed?
  - Detect impacted blocks (new & old changed)
- What can be leveraged?
  - Detect impacted blocks likely to be covered by existing tests
- What order should tests be run?
  - Detect minimal set of test cases likely to cover the impacted blocks

Computing Coverage

- Computed for each test T
- Old block b
  - Covered if T covered b in old binary
- New block
  - Covered if at least one predecessor and successor were covered in old binary
  - Heuristic: predict edges taken
  - Heuristic: don’t check predecessors for indirect call targets

Prioritization Algorithm

<table>
<thead>
<tr>
<th>Test</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>b2,b7</td>
</tr>
<tr>
<td>t2</td>
<td>b1,b2,b3,b8</td>
</tr>
<tr>
<td>t3</td>
<td>b7</td>
</tr>
<tr>
<td>t4</td>
<td>b6</td>
</tr>
<tr>
<td>t5</td>
<td>b1,b2,b5</td>
</tr>
<tr>
<td>t6</td>
<td>b4,b5</td>
</tr>
</tbody>
</table>

Impacted: b1,b2,b4,b7,b8
Echelon Performance:
ProductX.EXE

<table>
<thead>
<tr>
<th>Image Info</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build 2411.1</td>
<td>Build 2529.0</td>
</tr>
<tr>
<td>Date</td>
<td>Impacted Blocks</td>
</tr>
<tr>
<td>12/11/2000</td>
<td>378 (220 New, 158 Old)</td>
</tr>
<tr>
<td>Traces</td>
<td>Likely Covered by existing tests (LC)</td>
</tr>
<tr>
<td>31,020</td>
<td>176 Blocks</td>
</tr>
<tr>
<td>Blocks</td>
<td>Traces needed to cover LC (Set 1)</td>
</tr>
<tr>
<td>31,026</td>
<td>16 Traces</td>
</tr>
<tr>
<td>File size</td>
<td>Number of Traces in prioritized list</td>
</tr>
<tr>
<td>8,680,128</td>
<td>3,128</td>
</tr>
<tr>
<td>PDE size</td>
<td>Number of Traces</td>
</tr>
<tr>
<td>23,983,702</td>
<td>3,128</td>
</tr>
<tr>
<td>Number of Traces</td>
<td>Traces needed to cover LC (Set 1)</td>
</tr>
<tr>
<td>3,128</td>
<td>1,225</td>
</tr>
</tbody>
</table>

1.8 million lines of source code
Scout took about 210 seconds

Test Sequence Characteristics

Scout took about 210 seconds

Prediction Errors

1-4% False Positives
4-5% False Negatives

Defect Detection

Summary: Test Prioritization

- Effectively being used in MS Windows, SQL, and Exchange development process
  - Quickly identifies tests most likely to detect errors
- Scales to production environments - millions of tests and thousands of binaries
- Combination of approximations and static analysis to eliminate manual methods
- Collect information about development process