Efficient and Precise Dynamic Impact Analysis Using Execute-After Sequences

Taweesup (Term) Apiwattanapong

Alessandro Orso
Mary Jean Harrold

College of Computing
Georgia Institute of Technology

National Science Foundation awards CCR-0306372, CCR-0205422, CCR-0209322, SBE-0123532, and EIA-0196145 to Georgia Tech, Tata Consultancy Services
Impact analysis results

1. Develop P
2. Modify P
3. Develop T for P
4. Select subset of T to rerun
5. Identify faults
6. P correct for T?
7. T adequate?
8. Release P
9. Augment T for untested parts of P
Outline

• Dynamic Impact Analysis
• Existing Techniques
• Our Technique
• Empirical Studies
• Related Work
• Conclusion
Impact Analysis

Change = \{C\}

**Static Impact Analysis**
Impact Set = \{M, A, B, C, D\}

**Dynamic Impact Analysis**
Impact Set = \{M, A, B, C\}
Dynamic Impact Analysis

Impact-analysis techniques that
• Are based on dynamic information (e.g., test suites, field executions)
• Are conservative w.r.t. dynamic information

Quality of the dynamic information is key!
• representativeness of actual usage
  → collect actual usage → efficiency is important

Two existing dynamic impact-analysis techniques
• PathImpact
• CoverageImpact
Existing Techniques

Change = {C}

PathImpact

Impact Set = {M, B, C}

CoverageImpact

Impact Set = {M, A, B, C}


**Execute-After (EA) Relation**

**Essential information is**

“Execute-After Relation”

**Definition**

Given a program $P$, a set of executions $E$, and two methods $X$ and $Y$ in $P$, 

$$(X, Y) \in EA \text{ for } E \text{ if and only if, in at least one execution in } E,$$

1. $Y$ calls $X$ (directly or transitively (d/t)),

2. $Y$ returns into $X$ (d/t), or

3. $Y$ returns into a method $Z$ (d/t), and $Z$ later calls $X$ (d/t).
Computing EA Relation

Use method-entry and method-return-into events

\[ M_e \ A_e \ M_i \ \ldots \ A_e \ M_i \ B_e \ C_e \ B_i \ M_i \ B_e \]

Use only the first and last events of each method

\[ M_e \ A_e \ \ A_e \ B_e \ C_e \ M_i \ B_e \]

Execute-After (EA) Sequence

\[ M_f \ A_f \ \ A_l \ B_f \ C_f \ C_l \ M_l \ B_l \]
Our Technique

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Current Timestamp: 3
Our Technique

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

Current Timestamp: 10
Our Technique

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

Current Timestamp: 15

Sort the entries to obtain an Execute-After (EA) Sequence

M_f A_f A_l B_f C_f C_l M_l B_l
## Analytical Complexity

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Space</th>
<th>Time (per method call)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathImpact</td>
<td>$O(t)$</td>
<td>$O(t)$</td>
</tr>
<tr>
<td>CoverageImpact</td>
<td>$m$ bits</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Execute-After</td>
<td>$2m$ integers</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

$t$ = the size of the trace  
$m$ = the number of methods

Program size: 30 KLOC  
Trace size: 2 GB
Outline

• Dynamic Impact Analysis
• Existing Techniques
• Our Technique
• Empirical Studies
• Related Work
• Conclusion
Empirical Studies

Studies

– Efficiency
– Precision

Experimental Setup

Tool: EAT (Execute-After Tool)

Subject Programs:

<table>
<thead>
<tr>
<th>Program</th>
<th>Versions</th>
<th>Classes</th>
<th>Methods</th>
<th>LOC</th>
<th>Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siena</td>
<td>8</td>
<td>24</td>
<td>219</td>
<td>3674</td>
<td>564</td>
</tr>
<tr>
<td>Jaba</td>
<td>11</td>
<td>355</td>
<td>2695</td>
<td>33183</td>
<td>125</td>
</tr>
<tr>
<td>Jaba-long</td>
<td>11</td>
<td>355</td>
<td>2695</td>
<td>33183</td>
<td>90</td>
</tr>
</tbody>
</table>
EAT (Execute-After Tool)

Collecting method-return-into events

Normal return
Exceptional return into a catch block
Exceptional return into a finally block
Study 1: Efficiency

**Goal:** To evaluate relative execution costs for Execute-After (EA) wrt CoverageImpact (CI) and PathImpact (PI)

**Method:** Measure time to execute programs on test cases, gather dynamic data, and output information to disk.

<table>
<thead>
<tr>
<th>Program</th>
<th>Running time (ms)</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uninst. CI EA PI</td>
<td>CI EA PI</td>
</tr>
<tr>
<td>Siena</td>
<td>53 108 110 ~263</td>
<td>104 108 ~396</td>
</tr>
<tr>
<td>Jaba</td>
<td>432 463 486 ~54,000</td>
<td>7.18 12.50 ~12,400</td>
</tr>
<tr>
<td>Jaba-long</td>
<td>5257 5617 5861 -</td>
<td>6.85 11.49 -</td>
</tr>
</tbody>
</table>
Study 2: Precision

**Goal:** To compare the precision of the three techniques

**Method:** Measure the relative sizes of impact sets computed by the techniques.

50% smaller (250 methods)
Related Work

PathImpact (Law and Rothermel)
   based on lightweight dynamic forward slicing

CoverageImpact (Orso et al.)
   based on compressed program traces

Online impact analysis (Breech et al.)
   compute impact sets online
   space complexity: $x^2$
   time complexity (per method call): $O(x)$
   ( $x$ is the number of methods executed)
Conclusion

Summary
• identify essential information for dynamic impact analysis
• present a new, efficient, and precise technique to collect and analyze that information
• present a set of empirical studies which show the efficiency and effectiveness of our technique

Future directions
• perform studies using the technique in the field
• perform client-analysis
• generalization of the technique
  – levels of granularity
  – programming languages
• apply the technique to other dynamic analyses
  – reverse-engineering
  – recovery of feature interaction
Thank you.

Questions?