A FRAMEWORK FOR COST-EFFECTIVE DEPENDENCE-BASED DYNAMIC IMPACT ANALYSIS

Haipeng Cai and Raul Santelices
Department of Computer Science and Engineering
University of Notre Dame

SANER 2015 Supported by ONR Award N000141410037
Background

**Predictive DDIA**

**Dependence-based Dynamic Impact Analysis**

- **Program** (Base version of program)
- **Execution set** (Set of program inputs)
- **Query** (Potential change location)
- **Impact set** (Set of potential impacts)

2 Background
Problem

- Efficient approaches are too imprecise (e.g., PathImpact/EAS [T. Apiwattanapong et al., 2005])
- Precise approaches are too expensive (e.g., dynamic slicing [X. Zhang et al., 2004])
- Developers need techniques of multiple levels of cost-effectiveness tradeoffs for diverse needs (e.g., budgets versus the level of precision needed) [C.R. Souza et al., 2008]
**Approach**

- Utilize static dependencies in collaboration with method-level execution traces (i.e., hybrid approach)
- Exploit additional dynamic information
  - Statement coverage
  - Dynamic points-to data [M. Mock et al., 2005]
- Guide trace-based impact computation with both static and dynamic information
Solution

- A framework that unifies analysis techniques of various cost-effectiveness tradeoffs
  - Including existing representative options (PI/EAS)
  - Spawning three new instances

- Three new instances
  - TR: static dependencies + method TRaces
  - TC: TR + statement Coverage
  - FI: Full Information -- TC + dynamic points-to data
The Framework

Static Analysis

1. Compute static dependencies
2. Instrument for method event tracing
3. Instrument for coverage monitoring
4. Instrument for points-to monitoring
5. Traces

Runtime

1. Dependence graph
2. Dependence graph + traces
3. Dep. graph + traces + stmt. coverage
4. Dep. graph + traces + stmt. cov. + points-to
5. Traces

Post-processing

Query set $M$

Impact set of $M$
Algorithm

Dep. graph

Method trace

enter M5

return into M2

Dep. graph

Prune

TR Report

TC Report

Dep. graph

Stmt. coverage

2, 12, 13, 25, 145, …

Prune

Dyn. alias data

P1: O1, O2, O5...
P2: 02, O3
......

Prune

Dep. graph

FI Report

P1: O1, O2, O5...
P2: 02, O3
......
Experimental setup

- **Subjects**
  - 7 Java programs
  - Up to 212 KLOC in size (1k ~ 100k)

- **Techniques**
  - PI/EAS (baseline), TR, TC, FI (, FI+)

- **Metrics**
  - Effectiveness
    - Impact-set size ratios to baseline
  - Cost
    - Computation time
    - Storage space
  - Average cost-effectiveness
    - Percentage of impact-set reduction factor of time cost increase
Research questions

- How do the techniques compare in terms of effectiveness?
- How do the techniques compare in terms of costs?
- What are the effects of different forms of dynamic data on the DDIA cost-effectiveness?
Result: effectiveness

Effectiveness (Impact-set size ratio)

NanoXML

Ant

XML-security

TR vs PI/EAS
TC vs PI/EAS
FI vs PI/EAS
FI+ vs PI/EAS
Result: effectiveness

![Diagram showing effectiveness (Impact-set size ratio) for JMeter, Jaba, and ArgoUML with different comparisons (TR vs PI/EAS, TC vs PI/EAS, FI vs PI/EAS, FI+ vs PI/EAS).]
Research questions

- How do the techniques compare in terms of effectiveness?
- How do the techniques compare in terms of costs?
- What are the effects of different forms of dynamic data on the DDIA cost-effectiveness?
## Result: querying cost

<table>
<thead>
<tr>
<th>Subject</th>
<th>PI/EAS (seconds)</th>
<th>Query time of our techniques (seconds)</th>
<th>TR</th>
<th>TC</th>
<th>FI</th>
<th>FI+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule1</td>
<td>0.70</td>
<td>TR: 14.60, TC: 15.72, FI: 19.24, FI+: 44.26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NanoXML</td>
<td>0.07</td>
<td>TR: 6.24, TC: 6.35, FI: 5.60, FI+: 7.97</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML-security</td>
<td>0.04</td>
<td>TR: 7.43, TC: 8.01, FI: 8.15, FI+: 16.89</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMeter</td>
<td>0.02</td>
<td>TR: 2.25, TC: 2.30, FI: 1.82, FI+: 2.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ant-v0</td>
<td>0.05</td>
<td>TR: 3.19, TC: 3.39, FI: 3.31, FI+: 5.24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jaba</td>
<td>0.29</td>
<td>TR: 78.34, TC: 99.68, FI: 82.55, FI+: 105.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ArgoUML</td>
<td>0.05</td>
<td>TR: 15.95, TC: 15.98, FI: 12.60, FI+: 15.82</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>0.11</td>
<td>TR: 26.33, TC: 31.96, FI: 26.62, FI+: 35.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Result: other costs

### Static-analysis costs in seconds

<table>
<thead>
<tr>
<th>Subject</th>
<th>PI/EAS</th>
<th>TR</th>
<th>TC</th>
<th>FI/FI+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule 1</td>
<td>5</td>
<td>6</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>NanoXML</td>
<td>11</td>
<td>14</td>
<td>25</td>
<td>39</td>
</tr>
<tr>
<td>Ant</td>
<td>27</td>
<td>142</td>
<td>170</td>
<td>311</td>
</tr>
<tr>
<td>XML-security</td>
<td>33</td>
<td>158</td>
<td>190</td>
<td>280</td>
</tr>
<tr>
<td>JMeter</td>
<td>38</td>
<td>372</td>
<td>408</td>
<td>764</td>
</tr>
<tr>
<td>Jaba</td>
<td>55</td>
<td>289</td>
<td>326</td>
<td>600</td>
</tr>
<tr>
<td>ArgoUML</td>
<td>172</td>
<td>7,465</td>
<td>7,542</td>
<td>11,998</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td><strong>73</strong></td>
<td><strong>2,047</strong></td>
<td><strong>2,115</strong></td>
<td><strong>3,392</strong></td>
</tr>
</tbody>
</table>

- **Runtime costs: < 1m**
- **Space costs: < 4MB**
Research questions

- How do the techniques compare in terms of effectiveness?
- How do the techniques compare in terms of costs?
- What are the effects of different forms of dynamic data on the DDIA cost-effectiveness?
Result: cost-effectiveness

- With respect to querying costs

![Graph showing cost-effectiveness comparison between different tools]

- Schedule1
- NanoXML
- Ant
- XML-security
- JMeter
- Jaba
- ArgoUML
Result: cost-effectiveness

With respect to other costs

- Schedule 1
- NanoXML
- Ant
- XML-security
- JMeter
- Jaba
- ArgoUML
Conclusions

- A framework that unifies existing and new DDIA techniques, and offers multiple-level cost-effectiveness options
- New techniques greatly reducing impact-set sizes, implying large improvement in precision
- Statement coverage has generally stronger effects on DDIA cost-effectiveness than dynamic points-to data
Acknowledgements

Office of Naval Research for funding

All of you for time and attention
The proposed framework offers multiple-level trade-offs between cost and effectiveness of dynamic impact analysis.

Haipeng Cai
http://cse.nd.edu/~hcai/
hcai@nd.edu
## Subject programs

<table>
<thead>
<tr>
<th>Subject</th>
<th>KLOC</th>
<th>#Methods</th>
<th>#Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule1</td>
<td>0.3</td>
<td>24</td>
<td>2,650</td>
</tr>
<tr>
<td>NanoXml</td>
<td>3.5</td>
<td>282</td>
<td>214</td>
</tr>
<tr>
<td>Ant-v0</td>
<td>18.8</td>
<td>1,863</td>
<td>112</td>
</tr>
<tr>
<td>XML-security-v1</td>
<td>22.4</td>
<td>1,928</td>
<td>92</td>
</tr>
<tr>
<td>JMeter-v2</td>
<td>35.5</td>
<td>3,054</td>
<td>79</td>
</tr>
<tr>
<td>Jaba</td>
<td>37.9</td>
<td>3,332</td>
<td>70</td>
</tr>
<tr>
<td>ArgoUML-r3121</td>
<td>102.4</td>
<td>8,856</td>
<td>211</td>
</tr>
</tbody>
</table>
Achieving 100% recall with respect to actual impacts for dynamic dependence analysis is impossible.

Impact analysis is being emphasized all the time but practitioners mostly still stick to old-fashioned ways relying on manual efforts, what are possible obstacles there?
Key idea:

*Incrementally prune* methods NOT dependent on the query