Evolution-Aware Monitoring-Oriented Programming (eMOP)

Owolabi Legunsen, **Darko Marinov**, and Grigore Roşu

ICSE 2015 (NIER Track)
Florence, Italy
May 21, 2015
Monitoring-Oriented Programming (MOP)

Runtime monitoring of software against formal properties

• **Existing technique** targeted at single program version

Problems: High overhead and too many violations shown during evolution across many versions
Evolution-Aware MOP (eMOP)

Make MOP faster and show fewer violations during evolution

- **Proposed**

[Diagram showing the relationship between Code Changes, Property, Tests, MOP, Runtime Monitors, and Violation]
Input: (Potentially Buggy) Code

1  public boolean m(List a, List b) {
2      ...
3      for(Iterator i = a.iterator(); i.hasNext();){
4          ...
5      for(Iterator i2 = b.iterator(); i.hasNext();){
6          ... i2.next() ...
7      }
8      return ...
9  }

Line 5 should be  \textit{i2.hasNext()} 

Mimics two real bugs found in older AspectJ code
Input: Formally Specified Properties

1. When to fire Events
   
   after \texttt{Iterator.hasNext()} == \texttt{true}, before \texttt{Iterator.next()}

2. Specification over Events
   
   \texttt{Iterator.hasNext()} == \texttt{true} \textit{precedes} every \texttt{Iterator.next()}

3. Handler code
   
   User-defined action when specification is violated

Many properties can be monitored at once
public boolean find(List a, List b) {
    ...
    for (Iterator i = a.iterator(); i.hasNext();)
        ...
    for (Iterator i2 = b.iterator(); i2.hasNext();)
        // event: "before Iterator.next()"
        ...
    }
    return ...
}
Current State of MOP Research

- Many papers, focus on reducing runtime overhead
- Many bugs found in well-used, well-tested code
- All prior research focused on one version
  - Recurring costs of monitoring are high, e.g.,

<table>
<thead>
<tr>
<th>Run</th>
<th>Properties Monitored</th>
<th>Total Violations</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No MOP v1</td>
<td>n/a</td>
<td>n/a</td>
<td>8.4</td>
</tr>
<tr>
<td>MOP v1</td>
<td>180</td>
<td>27,895</td>
<td>164.1</td>
</tr>
<tr>
<td>MOP v2</td>
<td>180</td>
<td>27,904</td>
<td>231.8</td>
</tr>
</tbody>
</table>
Evolution-Aware MOP (eMOP)

• Improve MOP during software evolution
  • Faster: re-monitor based on parts affected by changes
  • Show fewer violations: show only violations due to changes

• We propose three techniques
  • Can be used separately or combined
  • Property selection
  • Monitor selection
  • Test selection
Technique: Property Selection

- What subset of properties to re-monitor in new version?
- Preliminary evaluation by seeding `i2.next()` bug:
  - Only `Iterator_HasNext` is affected by changes

<table>
<thead>
<tr>
<th>Run</th>
<th>Properties Monitored</th>
<th>Properties Violated</th>
<th>HasNext Violations</th>
<th>Total Violations</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No MOP v1</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>8.4</td>
</tr>
<tr>
<td>MOP v1</td>
<td>180</td>
<td>6</td>
<td>0</td>
<td>27,895</td>
<td>164.1</td>
</tr>
<tr>
<td>MOP v2</td>
<td>180</td>
<td>7</td>
<td>9</td>
<td>27,904</td>
<td>231.8</td>
</tr>
<tr>
<td>eMOP v2</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>9</td>
<td>8.8</td>
</tr>
</tbody>
</table>
Technique: Monitor Selection

• Generate monitors for parts of code affected by change
• Example: *Foo.java* and *Bar.java* both use Iterator
Technique: Test Selection (MOP + RTS)

• In eMOP we monitor execution of tests
  • RTS selects \textit{subset} of tests that can be affected by code changes
  • If fewer tests are run, fewer violations and less overhead
Some Challenges

• Safely determining properties/monitors/tests that can’t have new violations

• Non-determinism, e.g.,

In these versions, the same tests are run, but different number of violations
Conclusions

• All prior research on MOP targeted single code versions
• eMOP aims to adapt MOP to software evolution
  • Make MOP faster between versions of software
  • Show only violations due to changes between versions
• We proposed three techniques for eMOP
  • Property selection
  • Monitor selection
  • Test selection