An Empirical Investigation into the Impact of Refactoring on Regression Testing

Napol Rachatasumrit and Miryung Kim
The University of Texas at Austin
Motivation

- It is believed that refactoring improves software quality and maintainability
- Refactoring has the risk of functionality regression and increased testing cost
- The impact of refactoring edits on regression tests has not been investigated using version history.
We use **refactoring reconstruction analysis** and **change impact analysis** in tandem.

- Only 22% of refactored code is tested by existing regression tests.
- While refactoring edits constitute only 8% of changes, 38% of affected tests are relevant to refactorings.
- Refactoring edits appear in almost half of the failed test case execution.
Outline

- Motivation & Related Work
- Study Approach Overview
- Research Questions and Results
- Limitations
- Conclusions
Refactoring improves software quality and maintainability

A lack of refactoring incurs technical debt. [Cunningham, Lehman]

Refactor mercifully [Beck, eXtreme Programming]
The number of bug reports increases after refactorings [Weiβgerber & Diehl, Kim et al.]
Refactoring tools are buggy [Daniel et al.]
Programmers do not leverage refactoring tools effectively [Murphy-Hill et al. Vakilian et al.]
Refactoring comes with a risk of introducing subtle bugs and functionality regression [Kim et al.]
Outline

- Motivation
- **Study Approach Overview**
- Research Questions and Results
- Limitations
- Related Work
- Conclusions
Approach Overview

1. RefFinder: Refactoring Reconstruction
   [ICSM 2010, Prete et al.]
   Identify Refactoring Edits

2. FaultTracer: Change Impact Analysis
   [ICSM 2011, Zhang et al.]
   Identify Change Impact on Regression Tests

3. Investigate Refactoring Change Impact on Tests
### Step 1. Reconstruction of Refactoring Edits

**RefFinder: Refactoring Reconstruction**
[ICSM 2010, Prete et al.]

Identify edits that fit the program structure before and after each refactoring type

<table>
<thead>
<tr>
<th>Inferred Refactoring Edits: Type and Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>move_field(&quot;color&quot;, &quot;PieChart&quot;, &quot;Chart&quot;)</td>
</tr>
<tr>
<td>pull_up_field(&quot;color&quot;, &quot;PieChart&quot;, &quot;Chart&quot;)</td>
</tr>
<tr>
<td>collapse_hierarchy(&quot;Chart&quot;, &quot;PieChart&quot;)</td>
</tr>
<tr>
<td>introduce_explaining_var(&quot;val&quot;, &quot;EXPR…&quot;, &quot;get()&quot;)</td>
</tr>
</tbody>
</table>
We use our **logical program differencing framework**, LSdiff [ICSE 2009, Kim & Notkin] to compute change facts at the level of code elements, control and data dependences, etc.

<table>
<thead>
<tr>
<th>Original Fact-Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>past_subtype(“Chart”, “PieChart”), deleted_subtype(“Chart”, “PieChart”)</td>
</tr>
<tr>
<td>deleted_field(“PieChart.color”, “color”, “PieChart”)</td>
</tr>
<tr>
<td>added_field(“Chart.color”, “color”, “Chart”)</td>
</tr>
<tr>
<td>deleted_access(“PieChart.color”, “Chart.draw”), added_access(“Chart.color”, “Chart.draw”). . .</td>
</tr>
</tbody>
</table>
Step 1. Reconstruction of Refactoring Edits

We encode each refactoring type in a template logic rule.

<table>
<thead>
<tr>
<th>Refactoring Reconstruction Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. collapse hierarchy:</strong> A superclass and its subclass is not very different. Merge them together.</td>
</tr>
<tr>
<td>$(\text{deletedsubtype}(t_1,t_2) \land (\text{pull_up_field}(f,t_2,t_1) \lor \text{pull_up_method}(m,t_2,t_1))) \lor (\text{pastsubtype}(t_1,t_2) \land \text{deletedtype}(t_1,n,p) \land (\text{push_down_field}(f,t_1,t_2) \lor \text{push_down_method}(m,t_1,t_2)))) \Rightarrow \text{collapse_hierarchy}(t_1,t_2)$</td>
</tr>
<tr>
<td><strong>2. pull up method:</strong> A method is moved from a class to its superclass.</td>
</tr>
<tr>
<td>$\text{move_method}(f, t, t_1) \land \text{pastsubtype}(t_1, t) \Rightarrow \text{pull_up_method}(f, t, t_1)$</td>
</tr>
</tbody>
</table>
RefFinder converts the antecedent of a rule to a logic query and **invokes the query** on the change-fact database.

<table>
<thead>
<tr>
<th>Logic-Query Invocation</th>
<th>Added Facts</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleted_field(f₁, f, t₁) ^ added_field(f₂, f, t₂) ^ deleted _access(f₁, m₁) ^ added_access(f₂, m₁) ?</td>
<td>+ move_field(“color”, “PieChart”, “Chart”)</td>
</tr>
<tr>
<td>move_field(f, t₁, t₂) ^ past_subtype(t₂, t₁) ?</td>
<td>+ pull_up_field(“color”, “PieChart”, “Chart”)</td>
</tr>
<tr>
<td>invoking a collapse hierarchy query...</td>
<td>+ collapse_hierarchy(“Chart”, “PieChart”)</td>
</tr>
</tbody>
</table>
Step 2. Fault Tracer Change Impact Analysis

FaultTracer: Change Impact Analysis
[ICSM 2011, Zhang et al.]

affected tests—a set of regression tests relevant to atomic changes
affecting changes—a subset of atomic changes relevant to each affected test
Step 2. Fault Tracer Change Impact Analysis

Inputs:
Old Version P, Test Suite T, New Version P'
Step 2. Fault Tracer Change Impact Analysis

Inputs:
Old Version P, Test Suite T, New Version P’

Change Extraction
ECG (Extended Call Graph) Collection

Test 1
Method M1
Method M2
Field F1
Method M3

Test 2
Method M2
Method M3
Field F2
Method M3

…

calls
reads
calls
reads
Step 2. Fault Tracer Change Impact Analysis

Inputs:
Old Version P, Test Suite T, New Version P'

Change Extraction  ➔  ECG (Extended Call Graph) Collection  ➔  Affected Test Selection

Affected Tests: Test 1, …,
**Step 2. Fault Tracer Change Impact Analysis**

**Inputs:**
Old Version P, Test Suite T, New Version P'

- Change Extraction
- ECG (Extended Call Graph) Collection
- Affected Test Selection
- Affecting Change Determination

**Affecting Changes:**
- AF(F3), CM(M1), CM(M4), ...

**Test 1**
- Method M1
- Method M2
- Field F1
- Method M3

**AF F3**
- Method M4
  - calls to
  - Method M1
- calls to
  - Method M3
- reads from
  - Field F1

Cycles
- Method M3 calls Method M1
- Method M1 calls Method M2
- Method M2 reads Field F1
- Field F1 calls AF(F3)
Step 3. Refactoring Change Impact Assessment

Investigate Refactoring Change Impact on Tests

Identify Tests Affected by Refactoring Edits
Identify Refactoring Edits Affecting Tests

Refactored Elements

Refactoring Edits Affecting Tests:
Introduce_Local_Variable (M1), Move Field (F3), ...

Test 1
Method M1
AF
Method M4
F3
Method M2
Field F1
Method M3
## Data Sets

<table>
<thead>
<tr>
<th></th>
<th>JMeter</th>
<th>XMLSecurity</th>
<th>Ant</th>
</tr>
</thead>
<tbody>
<tr>
<td># Versions</td>
<td>6</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Releases</td>
<td>R0.0 to R5.0</td>
<td>R0.0 to R3.0</td>
<td>R0.0 to R8.0</td>
</tr>
<tr>
<td>LOC</td>
<td>31005~40695</td>
<td>17435~22863</td>
<td>17201~80444</td>
</tr>
<tr>
<td>Classes</td>
<td>313~402</td>
<td>181~154</td>
<td>172~650</td>
</tr>
<tr>
<td>Methods</td>
<td>2501~3237</td>
<td>1244~1023</td>
<td>1581~7190</td>
</tr>
<tr>
<td>Fields</td>
<td>830~970</td>
<td>129~151</td>
<td>440~3212</td>
</tr>
<tr>
<td>Refactoring Types</td>
<td>4~12</td>
<td>6~10</td>
<td>0~14</td>
</tr>
<tr>
<td>Total Correct Refactorings</td>
<td>349</td>
<td>161</td>
<td>511</td>
</tr>
<tr>
<td>Atomic Changes</td>
<td>307</td>
<td>214</td>
<td>1155</td>
</tr>
</tbody>
</table>
Outline

- Motivation and Related Work
- Study Approach Overview
- **Research Questions and Results**
- Limitations
- Conclusions
Research Questions

- Q1: Are there adequate tests for refactoring edits in practice?
- Q2: How many of existing regression tests are relevant to refactoring edits and thus need to be re-run for the new version?
- Q3: What proportion of failure-inducing changes are relevant to refactorings?
Q1. Are there adequate tests for refactoring edits in practice?

- **Test Coverage** \( \frac{|T|}{|A|} \)
  - The percentage of tested elements \(|T|\) out of all code elements \(|A|\)

- **Change Test Coverage** \( \frac{|T \cap C|}{|C|} \)
  - The percentage of changed elements exercised by tests \(|T \cap C|\) out of all changed elements \(|C|\)

- **Refactoring Test Coverage** \( \frac{|T \cap R|}{|R|} \)
  - The percentage of refactored elements exercised by tests \(|T \cap R|\) out of all refactored elements \(|R|\)
Q1. Are there adequate tests for refactoring edits in practice?

<table>
<thead>
<tr>
<th></th>
<th>Refactored Elements</th>
<th>Changed Elements</th>
<th>Tested Elements</th>
<th>Change Test Coverage</th>
<th>Refactoring Test Coverage</th>
<th>Test Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>R</td>
<td>$</td>
<td>$</td>
<td>C</td>
<td>$</td>
</tr>
<tr>
<td>JMeter</td>
<td>352</td>
<td>4040</td>
<td>5776</td>
<td>23.8%</td>
<td>16.5%</td>
<td>29.8%</td>
</tr>
<tr>
<td>XML</td>
<td>60</td>
<td>1101</td>
<td>1719</td>
<td>25.1%</td>
<td>61.7%</td>
<td>41.2%</td>
</tr>
<tr>
<td>Ant</td>
<td>326</td>
<td>4375</td>
<td>10588</td>
<td>15.1%</td>
<td>19.9%</td>
<td>25.7%</td>
</tr>
<tr>
<td>Total</td>
<td>738</td>
<td>9516</td>
<td>18038</td>
<td>19.9%</td>
<td>22.1%</td>
<td>27.9%</td>
</tr>
</tbody>
</table>

Only 22% of refactored methods and fields are tested by existing regression tests.
Q1. Are there adequate tests for refactoring edits in practice?

<table>
<thead>
<tr>
<th></th>
<th>Refactored Elements</th>
<th>Changed Elements</th>
<th>Tested Elements</th>
<th>Change Test Coverage</th>
<th>Refactoring Test Coverage</th>
<th>Test Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>R</td>
<td>$</td>
<td>$</td>
<td>C</td>
<td>$</td>
</tr>
<tr>
<td>JMeter</td>
<td>352</td>
<td>4040</td>
<td>5776</td>
<td>23.8%</td>
<td>16.5%</td>
<td>29.8%</td>
</tr>
<tr>
<td>XML</td>
<td>60</td>
<td>1101</td>
<td>1719</td>
<td>25.1%</td>
<td>61.7%</td>
<td>41.2%</td>
</tr>
<tr>
<td>Ant</td>
<td>326</td>
<td>4375</td>
<td>10588</td>
<td>15.1%</td>
<td>19.9%</td>
<td>25.7%</td>
</tr>
<tr>
<td>Total</td>
<td>738</td>
<td>9516</td>
<td>18038</td>
<td>19.9%</td>
<td>22.1%</td>
<td>27.9%</td>
</tr>
</tbody>
</table>

: If the refactoring edits are impure, more tests need to cover refactoring edits
Q2. How many of existing tests are relevant to refactoring edits?

- **AT**: affected tests
- **AT_R**: the ratio of affected tests that exercise at least one refactoring edit location
- **AC**: affecting changes
- **AC_R**: the ratio of affecting changes whose location overlaps with at least one refactoring edit
Q2. How many of existing tests are relevant to refactoring edits?

<table>
<thead>
<tr>
<th>Pair</th>
<th>Affected Tests</th>
<th>Tests Affected By Refactoring</th>
<th>Affecting Refactorings</th>
<th>Refactoring to Change Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>AT</td>
<td>)</td>
<td>(</td>
</tr>
<tr>
<td>JMeter</td>
<td>284</td>
<td>120 (42.2%)</td>
<td>70</td>
<td>8.7%</td>
</tr>
<tr>
<td>XML</td>
<td>180</td>
<td>133 (73.8%)</td>
<td>35</td>
<td>5.4%</td>
</tr>
<tr>
<td>Ant</td>
<td>1100</td>
<td>311 (28.2%)</td>
<td>85</td>
<td>7.4%</td>
</tr>
<tr>
<td>Total</td>
<td>1564</td>
<td>594 (38.0%)</td>
<td>190</td>
<td>7.8%</td>
</tr>
</tbody>
</table>

While refactoring edits constitute only 8% of atomic changes, 38% of affected tests are relevant to refactoring edits.
### Q2. How many of existing tests are relevant to refactoring edits?

<table>
<thead>
<tr>
<th>Pair</th>
<th>Affected Tests</th>
<th>Tests Affected By Refactoring</th>
<th>Affecting Refactorings</th>
<th>Refactoring to Change Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>AT</td>
<td>)</td>
<td>(</td>
</tr>
<tr>
<td>JMeter</td>
<td>284</td>
<td>120 (42.2%)</td>
<td>70</td>
<td>8.7%</td>
</tr>
<tr>
<td>XML</td>
<td>180</td>
<td>133 (73.8%)</td>
<td>35</td>
<td>5.4%</td>
</tr>
<tr>
<td>Ant</td>
<td>1100</td>
<td>311 (28.2%)</td>
<td>85</td>
<td>7.4%</td>
</tr>
<tr>
<td>Total</td>
<td>1564</td>
<td>594 (38.0%)</td>
<td>190</td>
<td>7.8%</td>
</tr>
</tbody>
</table>

If the refactorings are pure and can be isolated, then there’s an opportunity of saving testing cost.
Q3. How much of failure-inducing edits are related to refactorings?

- $\text{AT}_F$: affected tests that succeeded in the old version but failed in the new version
- $\text{AT}_{RF}$: a subset of $\text{AT}_F$ that exercise refactoring edits
- $\text{AC}_F$: failure-inducing changes, i.e., a set of affecting changes for the failed tests
- $\text{AC}_{RF}$: a subset of $\text{AC}_F$ that exercise the location of refactoring edits
**Q3. How much of failure-inducing edits are related to refactorings?**

| Pair     | Failed Affected Tests $| AT_F |$ | Tests Affected By Refactoring $| AT_RF |$ | Failure-Inducing Changes $| AC_F |$ | Failure-Inducing Refactorings $| AC_{RF} |$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JMeter</td>
<td>19</td>
<td>14</td>
<td>43</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td>5</td>
<td>5</td>
<td>12</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ant</td>
<td>61</td>
<td>20</td>
<td>607</td>
<td>57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>80</td>
<td>39</td>
<td>662</td>
<td>67</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Half of the failed affected tests include refactoring edits
Q3. How much of failure-inducing edits are related to refactorings?

<table>
<thead>
<tr>
<th>Pair</th>
<th>Failed Affected Tests</th>
<th>Tests Affected By Refactoring</th>
<th>Failure-Inducing Changes</th>
<th>Failure-Inducing Refactorings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>A_T^F</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>JMeter</td>
<td>19</td>
<td>14</td>
<td>43</td>
<td>3</td>
</tr>
<tr>
<td>XML</td>
<td>5</td>
<td>5</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>Ant</td>
<td>61</td>
<td>20</td>
<td>607</td>
<td>57</td>
</tr>
<tr>
<td>Total</td>
<td>80</td>
<td>39</td>
<td>662</td>
<td>67</td>
</tr>
</tbody>
</table>

Refactorings seem to appear on the execution traces of failed tests without being root failure causes.
Study Limitations and Future Work

- False negatives of refactoring reconstruction
- Our broader definition of refactoring edits—tolerating behavior modifications during our manual inspection
- Only three projects in SIR
- Our data is available in public:
  - http://users.ece.utexas.edu/~miryung/inspected_dataset.zip
We study the impact of refactoring edits on regression tests

- Refactoring test coverage is insufficient
- Though only a small portion of edits consists of refactoring edits, many tests are impacted by them

We need an automated regression test augmentation and validation approach targeting refactoring edits
This research is in part supported by National Science Foundation, CCF-1117902, CCF-1149391, and CCF-1043810 and Microsoft SEIF award.