Towards Refactoring-Aware Regression Test Selection

Kaiyuan Wang, Chenguang Zhu, Ahmet Celik, Jongwook Kim, Don Batory, Milos Gligoric

Funded in part by the US National Science Foundation and a Google Faculty Research Award
Regression Testing Is Important
Regression Testing Is Important
Regression Testing Is Important
Regression Testing is Costly

- Many companies report high cost
- Google Test Automation Platform (TAP) handles 800k builds and runs 150 million tests per day
- Microsoft’s CloudBuild is used by >4k developers and handles 20k builds per day
Regression Test Selection (RTS)

- Optimizes regression testing by skipping tests that are unaffected by recent code changes
- Maintains mapping from tests to code elements (statements, methods, classes)
- When code changes, uses mapping to find tests to run
- Many techniques developed over last 30 years, e.g.
  - TestTube (mapping from tests to functions)
  - FaultTracer (mapping from tests to methods)
  - Ekstazi (mapping from tests to classes)
  - HyRTS
Ekstazi Illustrated

- https://github.com/raphw/byte-buddy (Pull-up)

```java
abstract class AbstractBase {
    ...

class ForLoadedExecutable extends AbstractBase {
    @Override
    protected ParameterList wrap(List<ParameterDescription> values) {
        return new Explicit(values);
    }
}
```

```java
abstract class AbstractBase {
    @Override
    protected ParameterList wrap(List<ParameterDescription> values) {
        return new Explicit(values);
    }
}
```
Ekstazi Illustrated

- https://github.com/raphw/byte-buddy (Pull-up)

<table>
<thead>
<tr>
<th>JavaInstanceMethodTypeTest</th>
<th>JavaInstanceMethodHandleTest</th>
</tr>
</thead>
<tbody>
<tr>
<td>(AbstractBase, 7263), (ForLoadedExecutable, 4267), …</td>
<td>(AbstractBase, 7263), (ForLoadedExecutable, 4267), …</td>
</tr>
<tr>
<td>(AbstractBase, 1076), (ForLoadedExecutable, 1291), …</td>
<td>(AbstractBase, 1076), (ForLoadedExecutable, 1291), …</td>
</tr>
</tbody>
</table>

Ekstazi runs tests

| abstract class AbstractBase { |
| } ... |
| class ForLoadedExecutable extends AbstractBase { |
| @Override |
| protected ParameterList |
| wrap(List<ParameterDescription> values) { |
| return new Explicit(values); |
| } |
| } ... |
| class ForLoadedExecutable extends AbstractBase { |
| @Override |
| protected ParameterList |
| wrap(List<ParameterDescription> values) { |
| return new Explicit(values); |
| } |
| } ... |
How to Improve RTS for Refactorings

- Refactorings, i.e. behavior preserving changes, do not impact the test outcome
- Existing RTS techniques do not reason about semantics of changes and thus run all tests affected by refactorings, e.g., rename method
- Recent work has shown that refactorings do happen in practice [Silva+FSE'16, Tsantalis+ICSE'17]
- How can we improve RTS for behavior preserving changes?
Reks: Refactoring-Aware RTS

- Integrate with refactoring engines and file tracking systems
- Keep track of the files affected by refactoring changes and files affected by non-refactoring changes from the last commit
- Defines rules to update dependencies for each test without running any test
### Reks Illustrated

- [https://github.com/raphw/byte-buddy](https://github.com/raphw/byte-buddy) (Pull-up)

```java
abstract class AbstractBase {
    ...
}

class ForLoadedExecutable extends AbstractBase {
    @Override
    protected ParameterList wrap(List<ParameterDescription> values) {
        return new Explicit(values);
    }
}

JavaInstanceMethodTypeTest
(AbstractBase, 7263), (ForLoadedExecutable, 4267), ...
JavaInstanceMethodHandleTest
(AbstractBase, 7263), (ForLoadedExecutable, 4267), ...
```

Reks skips tests

<table>
<thead>
<tr>
<th>35da279</th>
<th>f1dfb66</th>
</tr>
</thead>
</table>
| abstract class AbstractBase { } ...
| class ForLoadedExecutable extends AbstractBase {
|     @Override
|     protected ParameterList wrap(List<ParameterDescription> values) {
|         return new Explicit(values);
|     }
| } ...
| abstract class AbstractBase {
|     @Override
|     protected ParameterList wrap(List<ParameterDescription> values) {
|         return new Explicit(values);
|     }
| } ...
| class ForLoadedExecutable extends AbstractBase {
| } ...
| JavaInstanceMethodTypeTest
(AbstractBase, 1076), (ForLoadedExecutable, 1291), ...
| JavaInstanceMethodHandleTest
(AbstractBase, 1076), (ForLoadedExecutable, 1291), ...
Reks Update Rules

- Defined three update rules to update dependencies
- Supports all (27) refactorings available in Eclipse
- Rules
  - Modify class
  - Replace class
  - Move elements
- More formally defined in the paper
Reks Update Rule (1): Modify Class

- https://github.com/apache/commons-compress (Rename Field)

<table>
<thead>
<tr>
<th>eee4f61</th>
<th>3e45dc8</th>
</tr>
</thead>
</table>
| ```java
public class GzipCompressorOutputStream {
    private final byte[] buffer = new byte[512];
    private void deflate() throws IOException {
        int length = deflater.deflate(buffer, 0, buffer.length);
        if (length > 0) {
            out.write(buffer, 0, length);
        }
    }
}
``` | ```java
public class GzipCompressorOutputStream {
    private final byte[] deflateBuffer = new byte[512];
    private void deflate() throws IOException {
        int length = deflater.deflate(deflateBuffer, 0, deflateBuffer.length);
        if (length > 0) {
            out.write(deflateBuffer, 0, length);
        }
    }
``` |
Reks Update Rule (1): Modify Class

- https://github.com/apache/commons-compress (Rename Field)

---

Reks skips tests

<table>
<thead>
<tr>
<th>eee4f61</th>
<th>GZipTestCase (GzipCompressorOutputStream, 5482), …</th>
</tr>
</thead>
</table>
| public class GzipCompressorOutputStream {
  private final byte[] buffer = new byte[512];
  private void deflate() throws IOException {
    int length = deflater.deflate(buffer, 0,
    buffer.length);
    if (length > 0) {
      out.write(buffer, 0, length);
    }|
| public class GzipCompressorOutputStream {
  private final byte[] deflateBuffer = new byte[512];
  private void deflate() throws IOException {
    int length = deflater.deflate(deflateBuffer, 0,
    deflateBuffer.length);
    if (length > 0) {
      out.write(deflateBuffer, 0, length);
    }|
| 3e45dc8 | GZipTestCase (GzipCompressorOutputStream, 9402), … |
Reks Update Rule (2): Replace Class

- https://github.com/google/auto (Rename Class)

```java
class JavaTokenizer {...}
final class AbstractMethodExtractor {
  ImmutableListMultimap<String, String>
  abstractMethods(JavaTokenizer tokenizer, String packageName) {...}
}
...
```

```java
class EclipseHackTokenizer {...}
final class AbstractMethodExtractor {
  ImmutableListMultimap<String, String>
  abstractMethods(EclipseHackTokenizer tokenizer, String packageName) {...}
}
...
```
### Reks Update Rule (2): Replace Class

- **https://github.com/google/auto** (Rename Class)

#### Code Examples

<table>
<thead>
<tr>
<th>26eaf2f</th>
<th>75a9cee</th>
</tr>
</thead>
</table>
| `class JavaTokenizer {...}  
final class AbstractMethodExtractor {  
  ImmutableListMultimap<String, String> abstractMethods(JavaTokenizer tokenizer, String packageName) {...}  
}  
... | `class EclipseHackTokenizer {...}  
final class AbstractMethodExtractor {  
  ImmutableListMultimap<String, String> abstractMethods(EclipseHackTokenizer tokenizer, String packageName) {...}  
}  
... |

| AbstractMethodExtractorTest (JavaTokenizer, 2839), … | AbstractMethodExtractorTest (EclipseHackTokenizer, 8347), … | Reks skips tests |
Reks Update Rule (3): Move Element

- https://github.com/apache/commons-crypto (Move Field)

<table>
<thead>
<tr>
<th>eee4f61</th>
<th>3e45dc8</th>
</tr>
</thead>
</table>
| public class CryptoCipherFactory {  
  ...
}  
public class ConfigurationKeys {  
  public static final String  
  CIPHER_CLASSES_DEFAULT =  
  OpensslCipher.class.getName();  
  ...
} | public class CryptoCipherFactory {  
  public static final String  
  CIPHER_CLASSES_DEFAULT =  
  OpensslCipher.class.getName();  
}  
public class ConfigurationKeys {  
  ...
} |
### Reks Update Rule (3): Move Element

- https://github.com/apache/commons-crypto (Move Field)

```
public class CryptoCipherFactory {
    ...
}
public class ConfigurationKeys {
    public static final String CIPHER_CLASSES_DEFAULT = OpensslCipher.class.getName();
    ...
}
```

```
public class CryptoCipherFactory {
    public static final String CIPHER_CLASSES_DEFAULT = OpensslCipher.class.getName();
}
public class ConfigurationKeys {
    ...
}
```

```
new Reks skips tests
```

<table>
<thead>
<tr>
<th>eee4f61</th>
<th>3e45dc8</th>
</tr>
</thead>
</table>
| public class CryptoCipherFactory {  
|    ...
| public class ConfigurationKeys {  
|    public static final String CIPHER_CLASSES_DEFAULT = OpensslCipher.class.getName();  
|    ...
| }  
| CryptoCipherFactoryTest  
| (CryptoCipherFactory, 4901),  
| (ConfigurationKeys, 3782), ...
| public class CryptoCipherFactory {  
|    public static final String CIPHER_CLASSES_DEFAULT = OpensslCipher.class.getName();  
| }  
| public class ConfigurationKeys {  
|    ...
| }  
| CryptoCipherFactoryTest  
| (CryptoCipherFactory, 2170),  
| (ConfigurationKeys, 7492), ...
|
Evaluation: Research Questions

- **RQ1**: How many *tests* would have been *skipped* by Reks had it been used by *open-source developers*?
Evaluation: Research Questions

- **RQ1**: How many tests would have been skipped by Reks had it been used by open-source developers?
- **RQ2**: How many tests does Reks skip on average if refactorings are systematically performed?
Evaluation: Research Questions

- **RQ1**: How many tests would have been skipped by Reks had it been used by open-source developers?
- **RQ2**: How many tests does Reks skip on average if refactorings are systematically performed?
- **RQ3**: How many tests does Reks skip on average for various refactoring types (e.g. move method) if refactorings are systematically performed?
Evaluation: Research Questions

- **RQ1**: How many tests would have been skipped by Reks had it been used by open-source developers?
- **RQ2**: How many tests does Reks skip on average if refactorings are systematically performed?
- **RQ3**: How many tests does Reks skip on average for various refactoring types (e.g. move method) if refactorings are systematically performed?
- **RQ4**: What is the cost of Reks update rules and how does this cost compare to the test execution time?
RQ1 How many tests would have been skipped by Reks had it been used by open-source developers?

\[ S_{\text{ref}} = \frac{|T_{\text{ref}}|}{|T_{\text{available}}|} \times 100 \]

\[ S_{\text{ref}} = \frac{|\{t1,t2\}|}{|\{t1,t2,t3\}|} = \frac{2}{3} = 67\% \]
RQ1  How many tests would have been skipped by Reks had it been used by open-source developers?

- 100 real refactorings from 37 projects
  - Reks saves 33% tests on average
  - 92 pure refactoring changes
    - Save [0%, 100%] tests, 34% on average
  - 8 mixed changes (refactoring + non-refactoring)
    - Save [0%, 64%] tests, 11% on average

\[
S_{\text{ref}} = \frac{|T_{\text{ref}}|}{|T_{\text{available}}|} \times 100
\]
RQ2 How many tests does Reks skip on average if refactorings are systematically performed?


<table>
<thead>
<tr>
<th>Project</th>
<th>LOC</th>
<th>#Test Class</th>
<th>Coverage (%)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>instruction</td>
<td>method</td>
<td>class</td>
<td></td>
</tr>
<tr>
<td>Coll</td>
<td>60251</td>
<td>159</td>
<td>83</td>
<td>77</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>Config</td>
<td>64341</td>
<td>163</td>
<td>87</td>
<td>83</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>DBCP</td>
<td>20547</td>
<td>30</td>
<td>45</td>
<td>56</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>IO</td>
<td>29159</td>
<td>100</td>
<td>86</td>
<td>82</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>JClassmate</td>
<td>6797</td>
<td>34</td>
<td>94</td>
<td>89</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>JObjectDiff</td>
<td>9976</td>
<td>61</td>
<td>89</td>
<td>84</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>Lang</td>
<td>69014</td>
<td>134</td>
<td>94</td>
<td>89</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Net</td>
<td>26928</td>
<td>42</td>
<td>32</td>
<td>26</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>Pebble</td>
<td>13375</td>
<td>30</td>
<td>86</td>
<td>79</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>Stateless4J</td>
<td>1702</td>
<td>9</td>
<td>53</td>
<td>43</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td><strong>Avg</strong></td>
<td><strong>30.2k</strong></td>
<td><strong>76.1</strong></td>
<td><strong>74.9</strong></td>
<td><strong>70.8</strong></td>
<td><strong>88.5</strong></td>
<td></td>
</tr>
</tbody>
</table>
RQ2 How many tests does Reks skip on average if refactorings are systematically performed?

% of Tests Reks skips per project

<table>
<thead>
<tr>
<th>Project</th>
<th>Max</th>
<th>Med</th>
<th>Avg</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coll</td>
<td>48.4</td>
<td>1.3</td>
<td>3.3</td>
<td>6.5</td>
</tr>
<tr>
<td>Config</td>
<td>72.4</td>
<td>4.3</td>
<td>13.8</td>
<td>18.0</td>
</tr>
<tr>
<td>DBCP</td>
<td>89.7</td>
<td>20.7</td>
<td>26.6</td>
<td>22.3</td>
</tr>
<tr>
<td>IO</td>
<td>63.0</td>
<td>9.0</td>
<td>10.9</td>
<td>10.1</td>
</tr>
<tr>
<td>JClassmate</td>
<td>85.3</td>
<td>32.4</td>
<td>37.1</td>
<td>25.5</td>
</tr>
<tr>
<td>JObjectDiff</td>
<td>85.3</td>
<td>8.2</td>
<td>21.4</td>
<td>23.5</td>
</tr>
<tr>
<td>Lang</td>
<td>52.2</td>
<td>1.5</td>
<td>4.6</td>
<td>7.1</td>
</tr>
<tr>
<td>Net</td>
<td>38.1</td>
<td>2.4</td>
<td>4.3</td>
<td>7.4</td>
</tr>
<tr>
<td>Pebble</td>
<td>96.7</td>
<td>96.7</td>
<td>67.3</td>
<td>40.4</td>
</tr>
<tr>
<td>Stateless4J</td>
<td>77.8</td>
<td>11.1</td>
<td>22.8</td>
<td>19.9</td>
</tr>
</tbody>
</table>

Max 96.7 96.7 67.3 40.4
Avg 70.9 - 21.2 -
RQ3 How many tests does Reks skip on average for various refactoring types if refactorings are systematically performed?

% of Tests Reks skips per refactoring type

<table>
<thead>
<tr>
<th>Refactoring</th>
<th>Max</th>
<th>Avg</th>
<th>Std</th>
<th>Refactoring</th>
<th>Max</th>
<th>Avg</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rename Field</td>
<td>96.7</td>
<td>13.0</td>
<td>20.4</td>
<td>Extract Superclass</td>
<td>96.7</td>
<td>16.9</td>
<td>24.4</td>
</tr>
<tr>
<td>Rename Method</td>
<td>96.7</td>
<td>22.3</td>
<td>27.1</td>
<td>Extract Interface</td>
<td>96.7</td>
<td>9.6</td>
<td>18.6</td>
</tr>
<tr>
<td>Rename Local</td>
<td>96.7</td>
<td>0.7</td>
<td>2.9</td>
<td>Use Supertype</td>
<td>96.7</td>
<td>2.5</td>
<td>11.9</td>
</tr>
<tr>
<td>Move Method</td>
<td>96.7</td>
<td>13.7</td>
<td>13.9</td>
<td>Push down</td>
<td>96.7</td>
<td>2.2</td>
<td>10.7</td>
</tr>
<tr>
<td>Change Signature</td>
<td>96.7</td>
<td>23.7</td>
<td>29.1</td>
<td>Pull up</td>
<td>96.7</td>
<td>21.1</td>
<td>23.6</td>
</tr>
<tr>
<td>Extract Method</td>
<td>96.7</td>
<td>19.3</td>
<td>24.1</td>
<td>Extract Class</td>
<td>96.7</td>
<td>19.0</td>
<td>26.3</td>
</tr>
<tr>
<td>Extract Local</td>
<td>96.7</td>
<td>18.0</td>
<td>22.5</td>
<td>Introduce Param Obj.</td>
<td>96.7</td>
<td>27.0</td>
<td>32.4</td>
</tr>
<tr>
<td>Extract Constant</td>
<td>96.7</td>
<td>15.0</td>
<td>21.2</td>
<td>Introduce Indirection</td>
<td>96.7</td>
<td>23.6</td>
<td>27.7</td>
</tr>
<tr>
<td>Inline Constant</td>
<td>96.7</td>
<td>9.2</td>
<td>15.7</td>
<td>Introduce Factory</td>
<td>96.7</td>
<td>15.0</td>
<td>25.9</td>
</tr>
<tr>
<td>Inline Method</td>
<td>96.7</td>
<td>12.0</td>
<td>20.9</td>
<td>Introduce Parameter</td>
<td>96.7</td>
<td>9.3</td>
<td>18.3</td>
</tr>
<tr>
<td>Inline Local</td>
<td>96.7</td>
<td>18.5</td>
<td>28.2</td>
<td>Encapsulate Field</td>
<td>96.7</td>
<td>12.0</td>
<td>21.6</td>
</tr>
<tr>
<td>Convert Local to Field</td>
<td>96.7</td>
<td>18.1</td>
<td>28.0</td>
<td>Generalize Type</td>
<td>96.7</td>
<td>15.8</td>
<td>25.2</td>
</tr>
<tr>
<td>Convert Anonymous</td>
<td>96.7</td>
<td>22.2</td>
<td>28.8</td>
<td>Infer Generic Type Args</td>
<td>12.6</td>
<td>0.1</td>
<td>0.6</td>
</tr>
<tr>
<td>Move Type to New File</td>
<td>72.1</td>
<td>9.2</td>
<td>14.7</td>
<td>Max</td>
<td>96.7</td>
<td>27.0</td>
<td>32.4</td>
</tr>
</tbody>
</table>
RQ4 What is the cost of Reks update rules and how does this cost compare to the test execution time?

### Execution Time for Reks, RetestAll and Ekstazi

<table>
<thead>
<tr>
<th>Project</th>
<th>Reks [s]</th>
<th>All [s]</th>
<th>Ekstazi [s]</th>
<th>R/A [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coll</td>
<td>23.0</td>
<td>59.0</td>
<td>34.6</td>
<td>38.9</td>
</tr>
<tr>
<td>Config</td>
<td>14.3</td>
<td>54.1</td>
<td>33.8</td>
<td>26.5</td>
</tr>
<tr>
<td>DBCP</td>
<td>6.9</td>
<td>86.7</td>
<td>35.8</td>
<td>7.9</td>
</tr>
<tr>
<td>IO</td>
<td>4.3</td>
<td>132.7</td>
<td>11.9</td>
<td>3.3</td>
</tr>
<tr>
<td>JClassmate</td>
<td>2.9</td>
<td>3.5</td>
<td>4.2</td>
<td>82.8</td>
</tr>
<tr>
<td>JObjectDiff</td>
<td>19.9</td>
<td>35.6</td>
<td>35.7</td>
<td>56.0</td>
</tr>
<tr>
<td>Lang</td>
<td>14.9</td>
<td>43.5</td>
<td>17.9</td>
<td>34.2</td>
</tr>
<tr>
<td>Net</td>
<td>3.9</td>
<td>63.1</td>
<td>4.9</td>
<td>6.1</td>
</tr>
<tr>
<td>Pebble</td>
<td>1.2</td>
<td>6.2</td>
<td>5.9</td>
<td>19.4</td>
</tr>
<tr>
<td>Stateless4J</td>
<td>1.8</td>
<td>2.2</td>
<td>2.8</td>
<td>79.0</td>
</tr>
<tr>
<td>Max</td>
<td>23.0</td>
<td>132.7</td>
<td>35.8</td>
<td>82.8</td>
</tr>
<tr>
<td>Avg</td>
<td>9.3</td>
<td>48.7</td>
<td>18.7</td>
<td>35.4</td>
</tr>
</tbody>
</table>
Reks Assumptions

- Refactoring changes are behavior-preserving
- Any bugs introduced by the refactoring engine is captured by test runs in the post-submit phase
- Users perform refactorings using refactoring engine
Conclusion

- Reks is the first refactoring-aware RTS technique
- Reks saves 33% of tests for real refactoring relevant changes, 16% of tests for artificial refactorings
- Reks saves 64.6% of test time compared to RetestAll
- http://cozy.ece.utexas.edu/reks/

Kaiyuan Wang (kaiyuanw@utexas.edu)
Chenguang Zhu (cgzhu@utexas.edu)
Ahmet Celik (ahmetcelik@utexas.edu)
Jongwook Kim (jongwook@utexas.edu)
Don Batory (dsb@cs.utexas.edu)
Milos Gligoric (gligoric@utexas.edu)
Q & A