AUTOMATED UNIT TEST GENERATION DURING SOFTWARE DEVELOPMENT

A Controlled Experiment and Think-aloud Observations

ISSTA 2015

José Miguel Rojas
j.rojas@sheffield.ac.uk

Joint work with Gordon Fraser and Andrea Arcuri
“Testing is a widespread validation approach in industry, but it is still largely ad hoc, expensive, and unpredictably effective.”

“Testing is a widespread validation approach in industry, but it is still largely ad hoc, expensive, and unpredictably effective.”


“Test case generation has a strong impact on the effectiveness and efficiency of testing.”

“…one of the most active research topics in software testing for several decades, resulting in many different approaches and tools.”

The gap between research and practice

In the last two decades, considerable research has been conducted in software testing and debugging. Unfortunately most research innovations did not make into practice; and software development and testing in the real world still lag behind in tooling support and automation. In this talk, I will share my limited experience and lessons learned in taking our research ideas into commercial tools as well as deploying them in large companies. I will discuss a few assumptions that we often make in our research but can significantly limit their adoption in practice. Additionally, I will also present some open problems that I have observed from interacting with customers and understanding their typical software testing workflows.
BACK IN ISSTA 2013…

Test generation helps…

...only if the user understands the test cases and the class under test!

“Does automated white-box test generation really help software testers?,” G. Fraser, M. Staats, P. McMinn, A. Arcuri and F. Padberg
BACK IN ISSTA 2013...

Test generation helps...

ARE UNIT TEST GENERATION TOOLS HELPFUL TO DEVELOPERS WHILE THEY ARE CODING?

...only if the user understands the test cases and the class under test!

“Does automated white-box test generation really help software testers?,” G. Fraser, M. Staats, P. McMinn, A. Arcuri and F. Padberg
CODE COVERAGE
CODE COVERAGE
TIME SPENT ON TESTING
CODE COVERAGE

TIME SPENT ON TESTING

IMPLEMENTATION QUALITY
CONTROLLED EXPERIMENT
CONTROLLED EXPERIMENT

class FilterIterator {
...
}

Golden Implementation and Test Suite
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite

Class Template

```java
class FilterIterator {
    ...;
}
```

```java
class FilterIterator {
    /**
     * Description
     */
    public void remove() {
        // TODO
    }
}
```
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite

Class Template

Implementation and Test Suite
CONTROLLED EXPERIMENT
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite

Class Template

EVA SUITE

Manual

Implementation and Test Suite
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite

Class Template

1 hour

Implementation and Test Suite

Manual
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite

Class Template

1 hour

Manual

Implementation and Test Suite
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite

class FilterIterator {
...

Apache Commons
http://commons.apache.org/

}

Class Template

class FilterIterator {
/**
 * Description
 */
public void remove() {
 // TODO
}
}

EVASUITE

Manual

1 hour

Implementation and Test Suite

class FilterIterator {
...
}
class TestFilterIterator {
...
}
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite

Class Template

EVA-SUITE Manual

Implementation and Test Suite

1 hour

Manual

class FilterIterator {
    ...}

class FilterIterator {
    /**
     * Description
     */
    public void remove() {
        // TODO
    }
}

class FilterIterator {
    ...}

class TestFilterIterator {
    ...}
DOES USING **EVOSUITE** DURING SOFTWARE DEVELOPMENT LEAD TO TEST SUITES WITH HIGHER CODE COVERAGE?

**RQ 1**
CODE COVERAGE

participants’ test suites run on their own implementations

- FilterIterator: Assisted 63%, Manual 39%
- FixedOrderComparator: Assisted 38%, Manual 57%
- ListPopulation: Assisted 83%, Manual 26%
- PredicatedMap: Assisted 41%, Manual 50%
CODE COVERAGE

Times coverage was checked

<table>
<thead>
<tr>
<th>Category Axis</th>
<th>Assisted</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterIterator</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>FixedOrderComparator</td>
<td>1.9</td>
<td>9.6</td>
</tr>
<tr>
<td>ListPopulation</td>
<td>5.9</td>
<td>4</td>
</tr>
<tr>
<td>PredicatedMap</td>
<td>5.3</td>
<td>6.4</td>
</tr>
</tbody>
</table>
CODE COVERAGE
participant’s test suites run on their own implementations

ListPopulation

- 100%
- 75%
- 50%
- 25%
- 0%

Time (min)
0 10 20 30 40 50 60

Branch Coverage (%)
- Manual
- EvoSuite
- Assisted
CODE COVERAGE

participant’s test suites run on their own implementations

ListPopulation

Branch Coverage (%)

Time (min)

0%
25%
50%
75%
100%

Manual
EvoSuite
Assisted
CODE COVERAGE

participant’s test suites run on their own implementations

![Code Coverage Diagram]

- Manual
- EvoSuite
- Assisted

ListPopulation

Branch Coverage (%)

Time (min)

0% 25% 50% 75% 100%

0 10 20 30 40 50 60

Manual: Green
EvoSuite: Red
Assisted: Blue
Code Coverage

Participant's test suites run on their own implementations.
CODE COVERAGE

participants’ test suites run on golden implementations

<table>
<thead>
<tr>
<th>Code Entity</th>
<th>Assisted</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterIterator</td>
<td>41%</td>
<td>30%</td>
</tr>
<tr>
<td>FixedOrderComparator</td>
<td>35%</td>
<td>21%</td>
</tr>
<tr>
<td>ListPopulation</td>
<td>37%</td>
<td>28%</td>
</tr>
<tr>
<td>PredicatedMap</td>
<td>42%</td>
<td>50%</td>
</tr>
</tbody>
</table>
CODE COVERAGE
participant’s test suites run on golden implementations, over time

FilterIterator

ListPopulation

FixedOrderComparator

PredicatedMap

Branch Coverage over Time for Assisted, Manual, and EvoSuite-generated tests.
CODE COVERAGE

participant’s test suites run on golden implementations, over time

- Assisted
- Manual
- EvoSuite-generated

Coverage can be higher when using **EvoSuite**, depending on how the generated tests are used.
DOES USING EVOSUITE DURING SOFTWARE DEVELOPMENT LEAD TO DEVELOPERS SPENDING MORE OR LESS TIME ON TESTING?

RQ 2
TESTING EFFORT

Number of test runs

<table>
<thead>
<tr>
<th>Class</th>
<th>Assisted</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterIterator</td>
<td>7.8</td>
<td>13.7</td>
</tr>
<tr>
<td>FixedOrderComparator</td>
<td>7.2</td>
<td>12.9</td>
</tr>
<tr>
<td>ListPopulation</td>
<td>8.2</td>
<td>11</td>
</tr>
<tr>
<td>PredicatedMap</td>
<td>4.4</td>
<td>5.6</td>
</tr>
</tbody>
</table>
TESTING EFFORT

Minutes spent on testing

<table>
<thead>
<tr>
<th>Component</th>
<th>Assisted</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterIterator</td>
<td>18.5</td>
<td>20</td>
</tr>
<tr>
<td>FixedOrderComparator</td>
<td>9.3</td>
<td>15.8</td>
</tr>
<tr>
<td>ListPopulation</td>
<td>12.6</td>
<td>25</td>
</tr>
<tr>
<td>PredicatedMap</td>
<td>7.7</td>
<td>14.3</td>
</tr>
</tbody>
</table>
Using **EvoSuite** reduces the time spent on testing.
DOES USING **EVOSUITE** DURING SOFTWARE DEVELOPMENT LEAD TO SOFTWARE WITH FEWER BUGS?

**RQ 3**
IMPLEMENTATION QUALITY

Golden test suites run on participants’ implementations

Number of Failures + Errors

<table>
<thead>
<tr>
<th>Component</th>
<th>Assisted</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterIterator</td>
<td>6.1</td>
<td>6.3</td>
</tr>
<tr>
<td>FixedOrderComparator</td>
<td>4.2</td>
<td>4.3</td>
</tr>
<tr>
<td>ListPopulation</td>
<td>6.4</td>
<td>5.3</td>
</tr>
<tr>
<td>PredicatedMap</td>
<td>15.6</td>
<td>14.3</td>
</tr>
</tbody>
</table>

---

FilterIterator: 6.1 Failures + Errors
FixedOrderComparator: 4.2 Failures + Errors
ListPopulation: 6.4 Failures + Errors
PredicatedMap: 15.6 Failures + Errors
IMPLEMENTATION QUALITY
Golden test suites run on participants’ implementations

Using **EvoSuite** during development did not lead to better implementations.
DOES SPENDING MORE TIME WITH EVOSUITE AND ITS TESTS LEAD TO BETTER IMPLEMENTATIONS?
PRODUCTIVITY

Time spent with EvoSuite

<table>
<thead>
<tr>
<th>FilterIterator</th>
<th>FixedOrderComparator</th>
<th>ListPopulation</th>
<th>PredicatedMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.29</td>
<td>-0.03</td>
<td>0.02</td>
<td>-0.49</td>
</tr>
<tr>
<td>0.35</td>
<td>0.32</td>
<td>0.35</td>
<td></td>
</tr>
</tbody>
</table>

Correlation with number of failures plus errors

-0.50  -0.40  -0.30  -0.20  -0.10  0.00  0.10  0.20  0.30  0.40

Number of runs

Time spent on tests
Implementation quality improves the more time developers spend with **EvoSuite**-generated tests.
Using automated unit test generation does impact developers’ productivity,
Using automated unit test generation does impact developers’ productivity, but...
Using automated unit test generation does impact developers’ productivity, but...

...how to make the most out of unit test generation tools?
THINK ALOUD OBSERVATIONS


THINK ALOUD OBSERVATIONS


THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

Class Template

Implementatio and Test Suite

class FilterIterator {
    ...
}

// Description
// // TODO

public void remove() {
// TODO
}

// EVA SUITE

5

1

4
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

Class Template

EVSUITE: Implementation and Test Suite

5

I

4
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

Class Template

Implementation and Test Suite

2 hours
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

5

Class Template

2 hours

Evosuite

Implementation and Test Suite

4
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

class FilterIterator {
    ...
}

Evosuite

class FilterIterator {
    /**
     * Description
     */
    public void remove() {
        // TODO
    }
}

Implementation and Test Suite

class FilterIterator {
    ...
}

class TestFilterIterator {
    ...
}
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

class FilterIterator {
    ...
}

Apache Commons
http://commons.apache.org/

class FilterIterator {
    /**
     * Description
     */
    public void remove() {
        // TODO
    }
}

class TestFilterIterator {
    ...
}

Implementation and Test Suite

2 hours
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

```
class FilterIterator {
    ...
}
```

Apache Commons
http://commons.apache.org/

EvSuite

Class Template

```
class FilterIterator {
    /**
     * Description
     */
    public void remove() {
        // TODO
    }
}
```

Implementation and Test Suite

```
class FilterIterator {
    ...
}
```

```
class TestFilterIterator {
    ...
}
```

2 hours

5

1

4
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

Class Template

2 hours

Implementation and Test Suite

5

4
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

Class Template

Implementation and Test Suite

2 hours

class FilterIterator {
    ...
}

/**
 * Description
 */
public void remove() {
    // TODO
}

class TestFilterIterator {
    ...
}
THINK ALOUD OBSERVATIONS

Golden Implementation and Test Suite

Class Template

Implementation and Test Suite

5 people

1 hour

4 weeks
RESULTS

<table>
<thead>
<tr>
<th>FilterIterator</th>
<th>FixedOrderComparator</th>
<th>ListPopulation</th>
<th>PredicatedMap</th>
<th>PredicatedMap</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>94%</td>
<td>92%</td>
<td>97%</td>
<td>72%</td>
<td>95%</td>
</tr>
<tr>
<td>JUnit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>23</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>94%</td>
<td>92%</td>
<td>97%</td>
<td>72%</td>
<td>95%</td>
</tr>
<tr>
<td>100%</td>
<td>94%</td>
<td>100%</td>
<td>83%</td>
<td></td>
</tr>
</tbody>
</table>

Images: http://jozef89.deviantart.com/
RESULTS

**EVA**SUITE

- FilterIterator: 3
- FixedOrderComparator: 6
- **ListPopulation**: 2
- PredicatedMap: 2
- PredicatedMap: 5

**JUnit**

- 15
- 20
- **23**
- 7
- 10

Scores:
- 94%
- 97%
- 94% | 95%
- 100%
- 100%

Images: http://jozef89.deviantart.com/
RESULTS

FilterIterator  FixedOrderComparator  ListPopulation  PredicatedMap  PredicatedMap

EVA SUITE 3  6  2  2  5

JUnit 15  20  23  7  10

94%  97% | 72%  94%  100%  100%

Images: http://jozef89.deviantart.com/
LESSONS LEARNED
LESSONS LEARNED

• There are different approaches to testing and test generation tools should be adaptable to them
LESSONS LEARNED

• There are different approaches to testing and test generation tools should be adaptable to them

• Developers’ behaviour is often not driven by code coverage
LESSONS LEARNED

- There are different approaches to testing and test generation tools should be *adaptable* to them.
- Developers’ behaviour is often *not driven by code coverage*.
- **Readability** of generated unit tests is paramount.
LESSONS LEARNED

• There are different approaches to testing and test generation tools should be adaptable to them

• Developers’ behaviour is often not driven by code coverage

• Readability of generated unit tests is paramount

• Integration into development environments must be improved
LESSONS LEARNED

• There are different approaches to testing and test generation tools should be adaptable to them

• Developers’ behaviour is often not driven by code coverage

• Readability of generated unit tests is paramount

• Integration into development environments must be improved

• Education/Best practices: Developers do not know how to best use automated test generation tools!
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite → Class Template → Manual → Implementation and Test Suite
CONTROLLED EXPERIMENT

Golden Implementation and Test Suite → Class Template → Manual → Implementation and Test Suite

1 hour

THINK ALOUD OBSERVATIONS

Observer

Subject


CONTROLLED EXPERIMENT

- Golden Implementation and Test Suite
- Class Template
- Implementation and Test Suite

1 hour

4

THINK ALOUD OBSERVATIONS


PRODUCTIVITY

Time spent with EvoSuite

Implementation quality improves the more time developers spend with EvoSuite-generated tests.
“Coverage is easy to assess because it is a number, while readability is a very non-tangible property…"
“Coverage is easy to assess because it is a number, while readability is a very non-tangible property…

… What is readable to me may not be readable to you. It is readable to me just because I spent the last hour and a half doing this.”
“Coverage is easy to assess because it is a number, while readability is a very non-tangible property…

… What is readable to me may not be readable to you. It is readable to me just because I spent the last hour and a half doing this.”

—Participant 5