Automated Support for Mobile Application Testing and Debugging

Mattia Fazzini

Georgia Tech College of Computing
Mobile Applications
Mobile Applications
Testing
Bug Reports

Arkonas commented on Jul 5, 2017

FC upon selecting weekly backup freq.
Steps to reproduce:
Settings -> Backup -> Backup freq.
100% for me, no other frequency setting.
Once after such crash, the app would not reproduce this.
Pretty sure the same thing happened.

Version: [Version information]
Phone: [Device information]
Android: [Android version information]
LinageOS: [LinageOS version information]
Kernel: [Kernel version information]

Trolldemorted commented on May 19, 2017

I have:
open and closed issues for:
https://github.com/WhisperSys

Description

Steps to reproduce

Onko commented on Mar 24, 2017

Steps to reproduce the behavior
1. Start a new post
2. Type something
3. Tap on the Publish button
4. The app crashes

Steps to reproduce

afwang commented on Mar 28, 2017

I found this crash on your:
redreader_crash_log

Reproduction steps:
1. Launch redreader in an un-subscribed sub
2. Tap the refresh button in the app bar
3. Swipe down while the subscribed sub is loaded

I think it might help to reproduce if your speed is throttled.
Goals

Help developers to find bugs early with novel testing techniques

Assist developers in quickly resolving bugs by automatically analyzing bug reports
Research

Test Generation

Differential Testing

Bug Report Analysis
Research

Test Generation

Differential Testing

Bug Report Analysis
Research

Test Generation

Differential Testing

Bug Report Analysis
Research

Test Generation

Differential Testing

Bug Report Analysis
Techniques

Test Generation

Barista

Differential Testing

DiffDroid

Bug Report Analysis

Yakusu
Research

Test Generation

Barista

Differential Testing

DiffDroid

Bug Report Analysis

Yakusu
Mobile App Testing

Challenges

- Not enough time to test: 52%
- Do not have the right testing process/method: 47%
- Do not have the right tools to test: 46%
- No mobile testing experts: 42%
- Do not have in-house testing environment: 41%
- Do not have the devices readily available: 40%
- We do not do testing: 5%
Mobile App Testing

Challenges

- Not enough time to test: 52%
- Do not have the right testing process/method: 47%
- Do not have the right tools to test: 46%
- No mobile testing experts: 42%
- Do not have in-house testing environment: 41%
- Do not have the devices readily available: 40%
- We do not do testing: 5%
Running Example: Manual Testing

Division by zero: nine divided by zero results in error message

(1) Press nine
(2) Press divide
(3) Press zero
(4) Press equal
(5) Check message
Running Example: Manual Testing

Division by zero: nine divided by zero results in error message

(1) Press nine
(2) Press divide
(3) Press zero
(4) Press equal
(5) Check message
Running Example: Manual Testing

Division by zero: nine divided by zero results in error message

(1) Press nine
(2) Press divide
(3) Press zero
(4) Press equal
(5) Check message

Simple
Running Example: Manual Testing

**Division by zero:** nine divided by zero results in error message

(1) Press nine  
(2) Press divide  
(3) Press zero  
(4) Press equal  
(5) Check message

---

Simple  
Inefficient, Tedious, Error prone
Running Example: Automated Testing

**Division by zero:** nine divided by zero results in error message

(1) Press nine
(2) Press divide
(3) Press zero
(4) Press equal
(5) Check message

```java
public void divisionByZero() {
    onView(withText("9")).perform(click());
    onView(withId(R.id.div)).perform(click());
    onView(withText("0")).perform(click());
    onView(withId(R.id.eq)).perform(click());
    onView(withId(R.id.dsp)).check(matches(withText("Error")));
}
```
Running Example: Automated Testing

Division by zero: nine divided by zero results in error message

(1) Press nine
(2) Press divide
(3) Press zero
(4) Press equal
(5) Check message

```java
public void divisionByZero() {
    onView(withText("9")) .perform(click());
    onView(withId(R.id.div)) .perform(click());
    onView(withText("0")) .perform(click());
    onView(withId(R.id.eq)) .perform(click());
    onView(withId(R.id.dsp))
      .check(matches(withText("Error")));
}
```
Running Example: Automated Testing

Division by zero: nine divided by zero results in error message

(1) Press nine
(2) Press divide
(3) Press zero
(4) Press equal
(5) Check message

```
public void divisionByZero() {
    onView(withText("9")).perform(click());
    onView(withId(R.id.div)).perform(click());
    onView(withText("0")).perform(click());
    onView(withId(R.id.eq)).perform(click());
    onView(withId(R.id.dsp))
        .check(matches(withText("Error")));
}
```
Division by zero: nine divided by zero results in error message

(1) Press nine
(2) Press divide
(3) Press zero
(4) Press equal
(5) Check message

```java
public void divisionByZero() {
    onView(withText("9")).perform(click());
    onView(withId(R.id.div)).perform(click());
    onView(withText("0")).perform(click());
    onView(withId(R.id.eq)).perform(click());
    onView(withId(R.id.dsp)).check(matches(withText("Error")));
}
```
Running Example: Automated Testing

Division by zero: nine divided by zero results in error message

(1) Press nine
(2) Press divide
(3) Press zero
(4) Press equal
(5) Check message

1 public void divisionByZero() {
2     onView(withText("9")).perform(click());
3     onView(withId(R.id.div)).perform(click());
4     onView(withText("0")).perform(click());
5     onView(withId(R.id.eq)).perform(click());
6     onView(withId(R.id.dsp))
7         .check(matches(withText("Error")));
8 }
Running Example: Barista

Manual Testing

Simplicity

Barista

Efficiency

Automated Testing

```java
public void divisionByZero() {
    8
}
```
Running Example: Barista

Manual Testing

Simplicity

Barista

Automated Testing

Efficiency

```java
public void divisionByZero() {
    // Code
}
```
Running Example: Barista

Manual Testing

Simplicity

Barista

Efficiency

Automated Testing

1 public void divisionByZero() {
    8 }

Running Example: Barista

Manual Testing

Simplicity

Barista

Efficiency

Automated Testing

```
public void divisionByZero() {
    onView(withXPath("/…"))
        .perform(click());
}
```
Running Example: Barista

1. `public void divisionByZero() {`
2. `onView(withXPath("/…")).perform(click());`
3. `onView(withId(R.id.div)).perform(click());`

8 `}`
Running Example: Barista

Manual Testing

Simplicity

Barista

Efficiency

Automated Testing

1 public void divisionByZero() {
2     onView(withXPath("/…")).perform(click());
3     onView(withId(R.id.div)).perform(click());
4     onView(withXPath("/…")).perform(click());
8 }

Running Example: Barista

Manual Testing

Simplicity

Efficiency

Automated Testing

1 public void divisionByZero() {
2     onView(withXPath("/..."));
3     onView(withId(R.id.div)).perform(click());
4     onView(withXPath("/..."));
5     onView(withId(R.id.eq)).perform(click());

8 }

Error

9 8 7 /
6 5 4 X
3 2 1 -
0 +
Running Example: Barista

Manual Testing                      Automated Testing

Simplicity                          Efficiency

1 public void divisionByZero() {
  2    onView(withXPath("/…")).perform(click());
  3    onView(withId(R.id.div)).perform(click());
  4    onView(withXPath("/…")).perform(click());
  5    onView(withId(R.id.eq)).perform(click());
  6    onView(withId(R.id.dsp))
  7       .check(matches(withText("Error")));
  8 }

Barista
Barista Overview

Application Under Test → Test Case Recording → Recorded Trace → Test Case Encoding → Test Case Execution → Test Report

Record Device

Test Devices
Barista Overview

Test Case Recording

Application Under Test → Recorded Trace → Recorded Trace

Recorded Device

Test Case Encoding

Test Case

Test Case Execution

Test Devices → Test Report
Barista Overview

Test Case Recording
- Application Under Test
- Record Device

Test Case Encoding
- Recorded Trace

Test Case Execution
- Test Case
- Test Devices
- Test Report
Test Case Recording

Objective: record actions performed on the AUT

Actions

Oracles
Test Case Recording

Objective: record actions performed on the AUT

Actions

Oracles
Test Case Recording

Objective: record actions performed on the AUT

Actions

Oracles
Test Case Recording: Elements
Test Case Recording: Elements

Accessibility

Actions
- click
- type
- scroll

Selector
- Relative Layout
  - Table Layout
    - TextView id=dsp
      - Button id=n text="9"
      - Button id=n text="8"
      - Button id=n text="7"
      - Button id=div text="/"
    - Table Row
      - Button id=n text="6"
      - Button id=n text="5"
      - Button id=n text="4"
      - Button id=mul text="X"
    - Table Row
      - Button id=n text="3"
      - Button id=n text="2"
      - Button id=n text="1"
      - Button id=sub text="-"
    - Table Row
      - Button id=n text="0"
      - Button id=eq text="="
      - Button id=add text="+

Interaction
- action
- selector
- props

Recorded Trace
Test Case Recording: Elements

Accessibility

Actions

click

type

scroll

...

Selector

Interaction

Recorded Trace

action

selector

props
Test Case Recording: Elements

Accessibility

Actions

click
type
scroll

Selector

Relative Layout

Table Layout

Table Row

Button id=n
text="9"

Button id=n
text="8"

Button id=n
text="7"

Button id=div
text="/"

Table Row

Button id=n
text="4"

Button id=n
text="5"

Button id=n
text="6"

Table Row

Button id=n
text="1"

Button id=n
text="2"

Button id=n
text="3"

Table Row

Button id=n
text="0"

Button id=eq
text="="

Button id=add
text="+"

Interaction

action
selector
props

Recorded Trace
Test Case Recording: Elements

Accessibility

Actions

Selector

Interaction

Accessibility

Actions

Selector

Interaction
Test Case Recording: Elements

Accessibility

Actions

- click
- type
- scroll

Selector

Interaction

Recorded Trace

```
Accessibility

Actions

- click
- type
- scroll

Selector

Interaction

Recorded Trace
```
Test Case Recording: Oracles
Test Case Recording: Oracles

Accessibility

Overlay

Selection

Assertion

Oracle

Recorded

Trace
Test Case Recording: Oracles
Test Case Recording: Oracles

Accessibility

Overlay

Selection

Assertion

Oracle

Recorded Trace

assertion

selector

props
Test Case Recording: Oracles

Accessibility

Overlay

Selection

Assertion

Oracle

assertion
selector
props

Recorded Trace
Test Case Recording: Oracles

Accessibility

Overlay

Selection

Assertion

Oracle

Recorded Trace
Test Case Recording: Oracles

Accessibility

Overlay

Selection

Assertion

Oracle

Recorded Trace
Test Case Encoding

Objective: translate recorded actions into a test case

Recorded Trace

Test Case

onView(withId(R.id.div)).perform(click());

onView(withId(R.id.dsp)).check(matches(withText("Error"));

onView(withId(R.id.dsp)).check(matches(withText("Error"));
Objective: translate recorded actions into a test case

Recorded Trace

Test Case

onView(withId(R.id.div)).perform(click());

onView(withId(R.id.dsp)).check(matches(withText("Error")));
Test Case Execution

Objective: execute test cases and generate a report

Test Devices

Test Case

Success
Error
Failure

Test Report
Test Case Execution

Objective: execute test cases and generate a report

Test Devices

Test Case

- Success
- Error
- Failure

Test Report
Test Case Execution

Objective: execute test cases and generate a report

Test Devices

Test Case

Success

Error

Failure

Test Report
Test Case Execution

Objective: execute test cases and generate a report

Test Devices

Test Case

Success

Error

Failure

Test Report
Empirical Evaluation

Research Questions:

RQ1: Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

RQ2: Is the test case recording process with BARISTA more efficient than the one with TR and ETR?

RQ3: Does BARISTA’s encoding preserve the functionality of the test cases? How does BARISTA compare to TR and ETR in this respect?

RQ4: Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?
Empirical Evaluation

Research Questions:

**RQ1:** Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

**RQ2:** Is the test case recording process with BARISTA more efficient than the one with TR and ETR?

**RQ3:** Does BARISTA’s encoding preserve the functionality of the test cases? How does BARISTA compare to TR and ETR in this respect?

**RQ4:** Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?
Empirical Evaluation

Research Questions:

RQ1: Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

RQ2: Is the test case recording process with BARISTA more efficient than the one with TR and ETR?

RQ3: Does BARISTA’s encoding preserve the functionality of the test cases? How does BARISTA compare to TR and ETR in this respect?

RQ4: Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?
Empirical Evaluation

Research Questions:

**RQ1:** Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

**RQ2:** Is the test case recording process with BARISTA more efficient than the one with TR and ETR?

**RQ3:** Does BARISTA’s encoding preserve the functionality of the test cases? How does BARISTA compare to TR and ETR in this respect?

**RQ4:** Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?
Empirical Evaluation

Research Questions:

**RQ1:** Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

**RQ2:** Is the test case recording process with BARISTA more efficient than the one with TR and ETR?

**RQ3:** Does BARISTA’s encoding preserve the functionality of the test cases? How does BARISTA compare to TR and ETR in this respect?

**RQ4:** Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?
Empirical Evaluation

Research Questions:

**RQ1:** Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

**RQ2:** Is the test case recording process with BARISTA more efficient than the one with TR and ETR?

**RQ3:** Does BARISTA’s encoding preserve the functionality of the test cases? How does BARISTA compare to TR and ETR in this respect?

**RQ4:** Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?
Study Protocol

First Task
- Apps
- NLTCs
- Test Cases
- x5, x5, x5

Second Task
- NLTCs
- Test Cases
- TR

Third Task
- NLTCs
- Test Cases
- ETR

Fourth Task
- NLTCs
- Test Cases
- BARISTA
Study Protocol

First Task

Apps


NLTCs

x5  x5  x5

Second Task

NLTCs

Test Cases

TR

Test Cases

Third Task

NLTCs

Test Cases

ETR

Test Cases

Fourth Task

NLTCs

Test Cases

BARISTA

Test Cases
Study Protocol

First Task
- Apps
- NLTCs
  - x5
  - x5
  - x5

Second Task
- NLTCs
  - TR
  - ETR
  - Test Cases

Third Task
- NLTCs
  - ETR
  - Test Cases

Fourth Task
- NLTCs
  - BARISTA
  - Test Cases
RQ1: Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>A1</td>
<td>15</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>A2</td>
<td>4</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>A3</td>
<td>9</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>A4</td>
<td>9</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>A5</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A6</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>A7</td>
<td>11</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>A8</td>
<td>11</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>A9</td>
<td>11</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>A10</td>
<td>13</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>A11</td>
<td>12</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>A12</td>
<td>15</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>A13</td>
<td>13</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>A14</td>
<td>15</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>A15</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Tot</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>
**RQ1**

**RQ1:** Can BARISTA record user-defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>A1</td>
<td>15</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>A2</td>
<td>4</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>A3</td>
<td>9</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>A4</td>
<td>9</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>A5</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A6</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>A7</td>
<td>11</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>A8</td>
<td>11</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>A9</td>
<td>11</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>A10</td>
<td>13</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>A11</td>
<td>12</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>A12</td>
<td>15</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>A13</td>
<td>13</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>A14</td>
<td>15</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>A15</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Tot</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>
**RQ1**

**RQ1:** Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR C(#)</th>
<th>TR NC(#)</th>
<th>TR AS(#)</th>
<th>TR AA(#)</th>
<th>ETR C(#)</th>
<th>ETR NC(#)</th>
<th>ETR AS(#)</th>
<th>ETR AA(#)</th>
<th>Barista C(#)</th>
<th>Barista NC(#)</th>
<th>Barista AS(#)</th>
<th>Barista AA(#)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>15</td>
<td>0</td>
<td>9</td>
<td>20</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>A2</td>
<td>4</td>
<td>11</td>
<td>0</td>
<td>2</td>
<td>15</td>
<td>0</td>
<td>14</td>
<td>3</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>A3</td>
<td>9</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>13</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>14</td>
<td>0</td>
<td>26</td>
<td>13</td>
</tr>
<tr>
<td>A4</td>
<td>9</td>
<td>5</td>
<td>8</td>
<td>7</td>
<td>14</td>
<td>0</td>
<td>26</td>
<td>13</td>
<td>14</td>
<td>0</td>
<td>26</td>
<td>13</td>
</tr>
<tr>
<td>A5</td>
<td>12</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>14</td>
<td>0</td>
<td>1</td>
<td>29</td>
<td>14</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>A6</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>A7</td>
<td>11</td>
<td>3</td>
<td>13</td>
<td>5</td>
<td>13</td>
<td>1</td>
<td>5</td>
<td>21</td>
<td>14</td>
<td>0</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>A8</td>
<td>11</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>14</td>
<td>0</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>A9</td>
<td>11</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>A10</td>
<td>13</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>14</td>
<td>1</td>
<td>17</td>
<td>8</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>A11</td>
<td>12</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>13</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>A12</td>
<td>15</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>4</td>
<td>14</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>A13</td>
<td>13</td>
<td>2</td>
<td>14</td>
<td>3</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>A14</td>
<td>15</td>
<td>0</td>
<td>7</td>
<td>2</td>
<td>11</td>
<td>4</td>
<td>1</td>
<td>14</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>A15</td>
<td>15</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>35</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**Tot** | **171** | **44** | **108** | **48** | **208** | **7** | **74** | **277** | **215** | **0** | **11** | **0**
### RQ1

**RQ1:** Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>A1</td>
<td>15</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>A2</td>
<td>4</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>A3</td>
<td>9</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>A4</td>
<td>9</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>A5</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A6</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>A7</td>
<td>11</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>A8</td>
<td>11</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>A9</td>
<td>11</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>A10</td>
<td>13</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>A11</td>
<td>12</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>A12</td>
<td>15</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>A13</td>
<td>13</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>A14</td>
<td>15</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>A15</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Tot</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>
**RQ1: Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?**

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>A1</td>
<td>15</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>A2</td>
<td>4</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>A3</td>
<td>9</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>A4</td>
<td>9</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>A5</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A6</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>A7</td>
<td>11</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>A8</td>
<td>11</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>A9</td>
<td>11</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>A10</td>
<td>13</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>A11</td>
<td>12</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>A12</td>
<td>15</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>A13</td>
<td>13</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>A14</td>
<td>15</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>A15</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Tot</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>
RQ1: Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>A15</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Tot</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>

100% completed tests for BARISTA

2% assertions skipped for BARISTA and none altered

More completed tests with BARISTA

More assertions skipped or altered with TR and ETR
RQ1: Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Tot</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>

**100% completed tests for BARISTA**

- 2% assertions skipped for BARISTA and none altered
- More completed tests with BARISTA
- More assertions skipped or altered with TR and ETR
**RQ1**

Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>A1</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>A2</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>

**100% completed tests for BARISTA**

2% assertions skipped for BARISTA and none altered

More completed tests with BARISTA

More assertions skipped or altered with TR and ETR
RQ1: Can BARISTA record user defined test cases? If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>A1</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>A2</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>

100% completed tests for BARISTA

2% assertions skipped for BARISTA and none altered

More completed tests with BARISTA

More assertions skipped or altered with TR and ETR
**RQ1: Can BARISTA record user defined test cases?** If so, how does it compare to Testdroid Recorder (TR) and Espresso Test Recorder (ETR)?

<table>
<thead>
<tr>
<th>ID</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C(#)</td>
<td>NC(#)</td>
<td>AS(#)</td>
</tr>
<tr>
<td>A15</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Tot</td>
<td>171</td>
<td>44</td>
<td>108</td>
</tr>
</tbody>
</table>

- **100% completed tests for BARISTA**
- **2% assertions skipped for BARISTA and none altered**
- **More completed tests with BARISTA**
- **More assertions skipped or altered with TR and ETR**
RQ4: Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?
RQ4: Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?

- LG G Flex
- Motorola Moto X
- HTC One M8
- Sony Xperia Z3
- Samsung Galaxy S5
- LG Nexus 5
- LG G3

TR: 68.3%
ETR: 37.3%
Barista: 99.2%
RQ4: Can test cases generated by BARISTA run on different devices? How does it compare to TR and ETR?

<table>
<thead>
<tr>
<th>Device</th>
<th>TR</th>
<th>ETR</th>
<th>Barista</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG G Flex</td>
<td>68.3%</td>
<td>37.3%</td>
<td>99.2%</td>
</tr>
<tr>
<td>Motorola Moto X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC One M8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sony Xperia Z3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy S5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG Nexus 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG G3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation Summary

**RQ1:** BARISTA can record test cases and is more expressive than TR and ETR

**RQ2:** BARISTA is more efficient in recording test cases than TR and ETR

**RQ3:** BARISTA generates test cases that work correctly and it outperforms TR and ETR

**RQ4:** BARISTA generates test cases that can run on different devices and it generates a greater number of cross-device-compatible tests than TR and ETR

**Feedback #1**

“I have been looking for something like BARISTA to help me get into automation for a while”

**Feedback #2**

“Overall, a very interesting tool! This could save us quite some time by generating some of the tests for us”

**Feedback #3**

“There are a few more assertions I’d like to see. For example, testing the number of items in a ListView”
Evaluation Summary

**RQ1:** BARISTA can record test cases and is more expressive than TR and ETR

**RQ2:** BARISTA is more efficient in recording test cases than TR and ETR

**RQ3:** BARISTA generates test cases that work correctly and it outperforms TR and ETR

**RQ4:** BARISTA generates test cases that can run on different devices and it generates a greater number of cross-device-compatible tests than TR and ETR

**Feedback #1**

“I have been looking for something like BARISTA to help me get into automation for a while”

**Feedback #2**

“Overall, a very interesting tool! This could save us quite some time by generating some of the tests for us”

**Feedback #3**

“There are a few more assertions I’d like to see. For example, testing the number of items in a ListView”
Evaluation Summary

**RQ1:** BARISTA can record test cases and is more expressive than TR and ETR

**RQ2:** BARISTA is more efficient in recording test cases than TR and ETR

**RQ3:** BARISTA generates test cases that work correctly and it outperforms TR and ETR

**RQ4:** BARISTA generates test cases that can run on different devices and it generates a greater number of cross-device-compatible tests than TR and ETR

---

**Feedback #1**

“I have been looking for something like BARISTA to help me get into automation for a while”

**Feedback #2**

“Overall, a very interesting tool! This could save us quite some time by generating some of the tests for us”

**Feedback #3**

“There are a few more assertions I’d like to see. For example, testing the number of items in a ListView”
Evaluation Summary

RQ1: BARISTA can record test cases and is more expressive than TR and ETR

RQ2: BARISTA is more efficient in recording test cases than TR and ETR

RQ3: BARISTA generates test cases that work correctly and it outperforms TR and ETR

RQ4: BARISTA generates test cases that can run on different devices and it generates a greater number of cross-device-compatible tests than TR and ETR

Feedback #1
“I have been looking for something like BARISTA to help me get into automation for a while”

Feedback #2
“Overall, a very interesting tool! This could save us quite some time by generating some of the tests for us”

Feedback #3
“There are a few more assertions I’d like to see. For example, testing the number of items in a ListView”
Evaluation Summary

RQ1: BARISTA can record test cases and is more expressive than TR and ETR

RQ2: BARISTA is more efficient in recording test cases than TR and ETR

RQ3: BARISTA generates test cases that work correctly and it outperforms TR and ETR

RQ4: BARISTA generates test cases that can run on different devices and it generates a greater number of cross-device-compatible tests than TR and ETR

Feedback #1
“I have been looking for something like BARISTA to help me get into automation for a while”

Feedback #2
“Overall, a very interesting tool! This could save us quite some time by generating some of the tests for us”

Feedback #3
“There are a few more assertions I’d like to see. For example, testing the number of items in a ListView”
Evaluation Summary

**RQ1:** **BARISTA** can record test cases and is more expressive than TR and ETR

**RQ2:** **BARISTA** is more efficient in recording test cases than TR and ETR

**RQ3:** **BARISTA** generates test cases that work correctly and it outperforms TR and ETR

**RQ4:** **BARISTA** generates test cases that can run on different devices and it generates a greater number of cross-device-compatible tests than TR and ETR

**Feedback #1**

“I have been looking for something like **BARISTA** to help me get into automation for a while”

**Feedback #2**

“Overall, a very interesting tool! This could save us quite some time by generating some of the tests for us”

**Feedback #3**

“There are a few more assertions I’d like to see. For example, testing the number of items in a ListView”
Evaluation Summary

RQ1: BARISTA can record test cases and is more expressive than TR and ETR

RQ2: BARISTA is more efficient in recording test cases than TR and ETR

RQ3: BARISTA generates test cases that work correctly and it outperforms TR and ETR

RQ4: BARISTA generates test cases that can run on different devices and it generates a greater number of cross-device-compatible tests than TR and ETR

Feedback #1
“I have been looking for something like BARISTA to help me get into automation for a while”

Feedback #2
“Overall, a very interesting tool! This could save us quite some time by generating some of the tests for us”

Feedback #3
“There are a few more assertions I’d like to see. For example, testing the number of items in a ListView”
Future Work

- Additional user study
- Reuse repetitive action sequences
- Sandboxing
- Extend assertable properties
- Fuzzing
- Fix broken test case during evolution
- Failure diagnosis
Future Work

- Additional user study
- Reuse repetitive action sequences
- Sandboxing
- Extend assertable properties
- Fuzzing
- Fix broken test case during evolution
- Failure diagnosis
Future Work

- Additional user study
- Reuse repetitive action sequences
- Sandboxing
- Extend assertable properties
- Fuzzing
- Fix broken test case during evolution
- Failure diagnosis
Future Work

- Additional user study
- Reuse repetitive action sequences
- Sandboxing
- Extend assertable properties
- Fuzzing
- Fix broken test case during evolution
- Failure diagnosis
Future Work

- Additional user study
- Reuse repetitive action sequences
- Sandboxing
- Extend assertable properties
- Fuzzing
- Fix broken test case during evolution
- Failure diagnosis
Future Work

- Additional user study
- Reuse repetitive action sequences
- Sandboxing
- Extend assertable properties
- Fuzzing
- Fix broken test case during evolution
- Failure diagnosis
Future Work

- Additional user study
- Reuse repetitive action sequences
- Sandboxing
- Extend assertable properties
- Fuzzing
- Fix broken test case during evolution
- Failure diagnosis
Future Work

- Additional user study
- Reuse repetitive action sequences
- Sandboxing
- Extend assertable properties
- Fuzzing
- Fix broken test case during evolution
- Failure diagnosis
Due to the wide array of available Android devices, we are targeting our support to a select number of Android devices to continue improving our overall Salesforce1 for Android user experience.
"Due to the wide array of available Android devices, we are targeting our support to a select number of Android devices to continue improving our overall Salesforce1 for Android user experience"
Due to the wide array of available Android devices, we are targeting our support to a select number of Android devices to continue improving our overall Salesforce1 for Android user experience.
”Due to the wide array of available Android devices, we are targeting our support to a select number of Android devices to continue improving our overall Salesforce1 for Android user experience”
"Due to the wide array of available Android devices, we are targeting our support to a select number of Android devices to continue improving our overall Salesforce1 for Android user experience"
Motivating Example

LG G3

LG Optimus L70
Motivating Example

LG G3

LG Optimus L70
Motivating Example

LG G3

LG Optimus L70
Motivating Example

LG G3

LG Optimus L70
Motivating Example

LG G3

LG Optimus L70

DiffDroid

CPI Report
DIFFDROID Overview

Reference Device → Input Generation

App Under Test → Trace

CPI Analysis → Reference UI Model

Test Case Encoding → Test Case

Test Case Execution → Test Devices

CPI Report
DIFFDROID Overview

Input Generation

Reference Device

App Under Test

CPI Analysis

Reference UI Model

CPI Report

Test Case Encoding

Trace

Test Case

Test Case Execution

Test Devices

Test Case

Input Generation

App

Reference Device

App Under Test

CPI Analysis

Reference UI Model

CPI Report

Test Case Encoding

Trace

Test Case

Test Case Execution

Test Devices

Test Case
DIFFDROID Overview

Reference Device

App Under Test

Input Generation

Test Case Encoding

Trace

CPI Analysis

Test Case Execution

Test Devices

CPI Report

Test UI Models

Test Cases
Input Generation

Reference Device

Inputs
Key
System
Touch

Window Model

UI Hierarchy

Screenshot
Input Generation

Window Model

Reference Device

Inputs

Key

System

Touch

UI Hierarchy

Screenshot
Input Generation

Reference Device

Inputs
- Key
- System
- Touch

Window Model

UI Hierarchy

Screenshot
Input Generation

Reference Device

Inputs
- Key
- System
- Touch

UI Hierarchy

Window Model

Screenshot
Input Generation

Reference Device

Inputs
- Key
- System
- Touch

Window Model

UI Hierarchy

Screenshot

Trace
Test Case Encoding

Trace

Test Case

Reference
UI Model
Test Case Encoding

Trace

Input

Touch

Test Case

Reference

UI Model
Test Case Encoding
Test Case Encoding

- **Trace**
  - Window Model
  - UI Hierarchy

- **Screenshot**
  - Screenshot
  - UI Model

- **Test Case**
  - Reference
Test Case Encoding

Window Model

UI Hierarchy

Screenshot

Test Case

Reference

UI Model

Trace

New
Test Case Encoding

UI Hierarchy

Screenshot

UI Model

Test Case

Trace

New

Reference
Test Case Execution

Test Case

Test Devices

Execution

Test UI Models
Test Case Execution

Test Case → Test Devices → Execution → Test UI Models
Test Case Execution

Test Case

Test Devices

Execution

Test UI Models
CPI Analysis
CPI Analysis

Reference Window Model

Test Window Model

Reference UI Hierarchy

Test UI Hierarchy

Reference Screenshot

Test Screenshot

Reference Node Matching

Test Node Matching

Reference Visual Analysis

Test Visual Analysis

Reference Structural Analysis

Test Structural Analysis

Reference CPI Report

Test CPI Report
CPI Analysis

Reference UI Model

Test UI Model

Reference Window Model

Test Window Model

Reference UI Hierarchy

Test UI Hierarchy

Reference Screenshot

Test Screenshot

Structural Analysis

Visual Analysis

Node Matching

CPI Report
CPI Analysis

Structural Analysis

Reference UI Hierarchy

Test UI Hierarchy

Reference Window Model

Test Window Model

Reference Screenshot

Test Screenshot

Node Matching

Visual Analysis

CPI Report
Structural Analysis

Node Similarity

Reference UI Hierarchy

Test UI Hierarchy

Resource ID

XPath

Properties

- checkable
- focusable
- clickable
- long-clickable
- scrollable

checked
focused
selected
enabled
text

Node Matching
Structural Analysis

Node Similarity

Resource ID

XPath

Properties

checkable  checked
focusable  focused
clickable  selected
long-clickable  enabled
scrollable  text
Structural Analysis

Node Similarity

Reference UI Hierarchy

Test UI Hierarchy

Resource ID

XPath

Properties

checkable  checked
focusable  focused
clickable  selected
long-clickable  enabled
scrollable  text

Node Matching
Structural Analysis

Node Similarity

Reference UI Hierarchy

Test UI Hierarchy

Resource ID

XPath

Properties

- checkable
- focusable
- clickable
- long-clickable
- scrollable

checked
focused
selected
enabled
text

Node Matching

Variable

Structural Inconsistency

CPI Report
Visual Analysis

Node Matching

Reference Screenshot

Test Screenshot

Relative Ratio Change (RRC)
Earth Mover Distance of Color Histogram (EMD)
Complex-Wavelet Structural Similarity Index (CW-SSIM)
Optical Character Recognition (OCR)

C4.5 Decision Tree Classifier
Visual Analysis

Node Matching

Reference Screenshot

Test Screenshot

Reference Node Image

Test Node Image

CW-SSIM  EMD  RRC  OCR

C4.5 Decision Tree Classifier

Complex-Wavelet Structural Similarity Index (CW-SSIM)

Earth Mover Distance of Color Histogram (EMD)

Relative Ratio Change (RRC)

Optical Character Recognition (OCR)
Visual Analysis

Node Matching

Reference Screenshot

Test Screenshot

Reference Node Image

Test Node Image

CW-SSIM  EMD  RRC  OCR

C4.5 Decision Tree Classifier

Complex-Wavelet Structural Similarity Index (CW-SSIM)

Earth Mover Distance of Color Histogram (EMD)

Relative Ratio Change (RRC)

Optical Character Recognition (OCR)
Visual Analysis

Node Matching

Reference Screenshot

Test Screenshot

Complex-Wavelet Structural Similarity Index (CW-SSIM)
Earth Mover Distance of Color Histogram (EMD)
Relative Ratio Change (RRC)
Optical Character Recognition (OCR)

C4.5 Decision Tree Classifier
Visual Analysis

Node Matching

Reference Screenshot

Test Screenshot

Reference Node Image

Test Node Image

CW-SSIM  EMD  RRC  OCR

C4.5 Decision Tree Classifier

→ Visual Inconsistency → CPI Report

Complex-Wavelet Structural Similarity Index (CW-SSIM)
Earth Mover Distance of Color Histogram (EMD)
Relative Ratio Change (RRC)
Optical Character Recognition (OCR)
Empirical Evaluation

Research Questions:

RQ1: Can DIFFDROID detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

RQ2: What is the cost of running DIFFDROID?

RQ3: Are there similarities among devices exhibiting CPIs?
Empirical Evaluation

Research Questions:

**RQ1:** Can DIFFDROID detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

**RQ2:** What is the cost of running DIFFDROID?

**RQ3:** Are there similarities among devices exhibiting CPIs?
Empirical Evaluation

Research Questions:

**RQ1:** Can DIFFDROID detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

**RQ2:** What is the cost of running DIFFDROID?

**RQ3:** Are there similarities among devices exhibiting CPIs?
Empirical Evaluation

Research Questions:

**RQ1:** Can DIFFDROID detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

**RQ2:** What is the cost of running DIFFDROID?

**RQ3:** Are there similarities among devices exhibiting CPIs?
Empirical Evaluation

Research Questions:

**RQ1:** Can DIFFDROID detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

**RQ2:** What is the cost of running DIFFDROID?

**RQ3:** Are there similarities among devices exhibiting CPIs?
# Benchmarks and Setup

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Category</th>
<th>Version</th>
<th>LOC (#K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>BuildmLearn</td>
<td>Education</td>
<td>2.5.0</td>
<td>23.6</td>
</tr>
<tr>
<td>A2</td>
<td>Daily Dozen</td>
<td>Health</td>
<td>10.3</td>
<td>6.3</td>
</tr>
<tr>
<td>A3</td>
<td>Kitchen Timer</td>
<td>Tools</td>
<td>1.1.6</td>
<td>4.3</td>
</tr>
<tr>
<td>A4</td>
<td>Outlay</td>
<td>Finance</td>
<td>1.1.3</td>
<td>8</td>
</tr>
<tr>
<td>A5</td>
<td>Translation Studio</td>
<td>Books</td>
<td>9.0</td>
<td>51.2</td>
</tr>
</tbody>
</table>

## Reference Device
LG G3, Android 22

## Test Devices

<table>
<thead>
<tr>
<th>Resolution</th>
<th>OS Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>720x1280</td>
<td>19</td>
</tr>
<tr>
<td>768x1280</td>
<td>21</td>
</tr>
<tr>
<td>1080x1920</td>
<td>22</td>
</tr>
<tr>
<td>1440x2560</td>
<td>23</td>
</tr>
<tr>
<td>480x800</td>
<td>24</td>
</tr>
<tr>
<td>540x960</td>
<td>25</td>
</tr>
<tr>
<td>480x854</td>
<td></td>
</tr>
</tbody>
</table>
# Benchmarks and Setup

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Category</th>
<th>Version</th>
<th>LOC (#K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>BuildmLearn</td>
<td>Education</td>
<td>2.5.0</td>
<td>23.6</td>
</tr>
<tr>
<td>A2</td>
<td>Daily Dozen</td>
<td>Health</td>
<td>10.3</td>
<td>6.3</td>
</tr>
<tr>
<td>A3</td>
<td>Kitchen Timer</td>
<td>Tools</td>
<td>1.1.6</td>
<td>4.3</td>
</tr>
<tr>
<td>A4</td>
<td>Outlay</td>
<td>Finance</td>
<td>1.1.3</td>
<td>8</td>
</tr>
<tr>
<td>A5</td>
<td>Translation Studio</td>
<td>Books</td>
<td>9.0</td>
<td>51.2</td>
</tr>
</tbody>
</table>

**Reference Device**

LG G3, Android 22

**Test Devices**

- Resolution
  - 720x1280
  - 768x1280
  - 1080x1920
  - 1440x2560
  - 480x800
  - 540x960
  - 480x854
- OS Version
  - 19
  - 21
  - 22
  - 23
  - 24
  - 25

147
## Benchmarks and Setup

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Category</th>
<th>Version</th>
<th>LOC (#K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>BuildmLearn</td>
<td>Education</td>
<td>2.5.0</td>
<td>23.6</td>
</tr>
<tr>
<td>A2</td>
<td>Daily Dozen</td>
<td>Health</td>
<td>10.3</td>
<td>6.3</td>
</tr>
<tr>
<td>A3</td>
<td>Kitchen Timer</td>
<td>Tools</td>
<td>1.1.6</td>
<td>4.3</td>
</tr>
<tr>
<td>A4</td>
<td>Outlay</td>
<td>Finance</td>
<td>1.1.3</td>
<td>8</td>
</tr>
<tr>
<td>A5</td>
<td>Translation Studio</td>
<td>Books</td>
<td>9.0</td>
<td>51.2</td>
</tr>
</tbody>
</table>

**Reference Device**

LG G3, Android 22

**Test Devices**

147

<table>
<thead>
<tr>
<th>Resolution</th>
<th>OS Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>720x1280</td>
<td>19</td>
</tr>
<tr>
<td>768x1280</td>
<td>21</td>
</tr>
<tr>
<td>1080x1920</td>
<td>22</td>
</tr>
<tr>
<td>1440x2560</td>
<td>23</td>
</tr>
<tr>
<td>480x800</td>
<td>24</td>
</tr>
<tr>
<td>540x960</td>
<td>25</td>
</tr>
<tr>
<td>480x854</td>
<td></td>
</tr>
</tbody>
</table>
**RQ1**

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

<table>
<thead>
<tr>
<th>ID</th>
<th>Test Device</th>
<th>Window Model</th>
<th>Structural CPI</th>
<th>Functional CPI</th>
<th>Version CPI</th>
<th>Cosmetic CPI</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>135</td>
<td>19</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>A2</td>
<td>138</td>
<td>22</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>A3</td>
<td>129</td>
<td>13</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>A4</td>
<td>125</td>
<td>14</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>A5</td>
<td>136</td>
<td>17</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>19</td>
<td>8</td>
</tr>
</tbody>
</table>

| Total | 6 | 9 | 7 | 74 | 16 |
**RQ1**

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

<table>
<thead>
<tr>
<th>ID</th>
<th>Test Device</th>
<th>Window Model</th>
<th>Structural CPI</th>
<th>Functional CPI</th>
<th>Version CPI</th>
<th>Cosmetic CPI</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>135</td>
<td>19</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>A2</td>
<td>138</td>
<td>22</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>A3</td>
<td>129</td>
<td>13</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>A4</td>
<td>125</td>
<td>14</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>A5</td>
<td>136</td>
<td>17</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>19</td>
<td>8</td>
</tr>
</tbody>
</table>

**Total:** 6 9 7 74 16
RQ1: Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

<table>
<thead>
<tr>
<th>ID</th>
<th>Test Device</th>
<th>Window Model</th>
<th>Structural CPI</th>
<th>Functional CPI</th>
<th>Version CPI</th>
<th>Cosmetic CPI</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>135</td>
<td>19</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>A2</td>
<td>138</td>
<td>22</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>A3</td>
<td>129</td>
<td>13</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>A4</td>
<td>125</td>
<td>14</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>A5</td>
<td>136</td>
<td>17</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>19</td>
<td>8</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>74</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**RQ1**

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

<table>
<thead>
<tr>
<th>Test Device</th>
<th>Structural CPI</th>
<th>Functional CPI</th>
<th>Version CPI</th>
<th>Cosmetic CPI</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>Example</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>74</td>
<td>16</td>
</tr>
</tbody>
</table>
**RQ1**

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

### Example

<table>
<thead>
<tr>
<th>Structural CPI</th>
<th>Functional CPI</th>
<th>Version CPI</th>
<th>Cosmetic CPI</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>7</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>7</td>
<td>74</td>
<td>16</td>
</tr>
</tbody>
</table>

Reference Device

Test Device

Start

Stop

Start
RQ1: Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?
**RQ1**

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

<table>
<thead>
<tr>
<th>Example</th>
<th>Reference Device</th>
<th>Test Device</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Structural CPI</th>
<th>Functional CPI</th>
<th>Version CPI</th>
<th>Cosmetic CPI</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>7</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>7</td>
<td>74</td>
<td>16</td>
</tr>
</tbody>
</table>
RQ1

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?
RQ1: Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?
RQ1: Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?
## Evaluation Summary

**RQ1: Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?**

*DIFFDROID* can detect CPIs in mobile applications while reporting a limited number of false positives.

**RQ2: What is the cost of running DiffDroid?**

*DIFFDROID* can run overnight for the cases considered.

**RQ3: Are there similarities among devices exhibiting CPIs?**

Devices that are more problematic have low values for resolution and density. However, considering testing devices solely based on resolution and density would have not allowed us to identify all the inconsistencies reported.
Evaluation Summary

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

**DIFFDROID** can detect CPIs in mobile applications while reporting a limited number of false positives.

**RQ2:** What is the cost of running DiffDroid?

**DIFFDROID** can run overnight for the cases considered.

**RQ3:** Are there similarities among devices exhibiting CPIs?

Devices that are more problematic have low values for resolution and density. However, considering testing devices solely based on resolution and density would have not allowed us to identify all the inconsistencies reported.
**Evaluation Summary**

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

**DIFFDROID** can detect CPIs in mobile applications while reporting a limited number of false positives.

**RQ2:** What is the cost of running DiffDroid?

**DIFFDROID** can run overnight for the cases considered.

**RQ3:** Are there similarities among devices exhibiting CPIs?

Devices that are more problematic have low values for resolution and density. However, considering testing devices solely based on resolution and density would have not allowed us to identify all the inconsistencies reported.
**Evaluation Summary**

**RQ1:** Can DiffDroid detect cross-platform inconsistencies in mobile applications while reporting a limited number of false positives?

**DIFFDROID** can detect CPIs in mobile applications while reporting a limited number of false positives.

**RQ2:** What is the cost of running DiffDroid?

**DIFFDROID** can run overnight for the cases considered.

**RQ3:** Are there similarities among devices exhibiting CPIs?

Devices that are more problematic have low values for resolution and density. However, considering testing devices solely based on resolution and density would have not allowed us to identify all the inconsistencies reported.
Future Work

- Additional user study
- API based differential analysis
- Multi-class classifier approach
- CPIs repair technique
Future Work

- Additional user study
- API based differential analysis
- Multi-class classifier approach
- CPIs repair technique
Future Work

- Additional user study
- API based differential analysis
- Multi-class classifier approach
- CPIs repair technique
Future Work

- Additional user study
- API based differential analysis
- Multi-class classifier approach
- CPIs repair technique
Future Work

- Additional user study
- API based differential analysis
- Multi-class classifier approach
- CPIs repair technique
Bug Reports

Arkconcs commented on Jul 5, 2017

FC upon selecting weekly backup feature.

Steps to reproduce:
Settings -> Backup -> Backup frequency
100% for me, no other frequency setting worked.

Once after such a crash, the app would not reproduce this.
Pretty sure the same thing happens.

Version: 5.2.3
Phone: Nexus 6
Android 7.1.2
LinageOS 14.1
Kernel: 3.10.65-1756-g8210f94a3d68

Trollidemorted commented on May 19, 2017

I have:

open and closed issues for:
https://github.com/WhisperSyst

Description:

Steps to reproduce:

Steps to reproduce the behavior:
1. Start a new post
2. Type something
3. Tap on the Publish button
4. The app crashes

Text paragraph:

<Logcat information>

Actual behavior:

Twidere crashes

Steps to reproduce:

Longpress retweet icon underneath t
Motivating Example

Bug Report

Onko commented on Mar 24, 2017

Steps to reproduce the behavior

1. Start a new post
2. Type something
3. Tap on the Publish button
4. The app crashes

<Text paragraph>

<Logcat information>

Developer

WORDPRESS App
Motivating Example

**Bug Report**

Onko commented on Mar 24, 2017

*Text paragraph>*

*Logcat information>*

**Steps to reproduce the behavior**

1. Start a new post
2. Type something
3. Tap on the Publish button
4. The app crashes

*Text paragraph>*

**WORDPRESS App**

Developer
Motivating Example

**Bug Report**

Onko commented on Mar 24, 2017

<Text paragraph>

<Logcat information>

**Steps to reproduce the behavior**

1. Start a new post
2. Type something
3. Tap on the Publish button
4. The app crashes

<Text paragraph>

**WORDPRESS App**

Developer

[Image of a WordPress app interface with text highlighted]

[Image of a user interface with text highlighted]

[Image of a user interface with text highlighted]

[Image of a user interface with text highlighted]
Motivating Example

Bug Report

Onko commented on Mar 24, 2017

<Text paragraph>

<Logcat information>

Steps to reproduce the behavior
1. Start a new post
2. Type something
3. Tap on the Publish button
4. The app crashes

<Text paragraph>

Developer

WORDPRESS App

GrboVK9

Content (tap to add text and media)
Motivating Example

**Bug Report**

Onko commented on Mar 24, 2017

*<Text paragraph>*

*<Logcat information>*

**Steps to reproduce the behavior**

1. Start a new post
2. Type something
3. Tap on the Publish button
4. The app crashes

*<Text paragraph>*

**WORDPRESS App**

Developer

Unfortunately, WordPress has stopped.

OK
Motivating Example

**WordPress App**

**Test Case**

```java
public void test() {
    // start a new post
    onView(
        withId(R.id.fab_button))
        .perform(click());

    // type something
    onView(
        withId(R.id.post_title))
        .perform(typeText("GrbcVK9I"));

    // tap on the publish button
    onView(
        withId(R.id.menu_save_post))
        .perform(click());
}
```
Motivating Example

**WORDPRESS App**

**Test Case**

```java
public void test() {
    // start a new post
    onView(
        withId(R.id.fab_button))
        .perform(click());

    // type something
    onView(
        withId(R.id.post_title))
        .perform(typeText("GrbcVK9I"));

    // tap on the publish button
    onView(
        withId(R.id.menu_save_post))
        .perform(click());
}
```
Motivating Example

**WORDPRESS App**

**Test Case**

```
public void test() {
    // start a new post
    onView(
       (withId(R.id.fab_button))
    ).perform(click());

    // type something
    onView(
       (withId(R.id.post_title))
    ).perform(typeText("GrbcVK9I"));

    // tap on the publish button
    onView(
       (withId(R.id.menu_save_post))
    ).perform(click());
}
```
Motivating Example

Bug Report

Onko commented on Mar 24, 2017

<Text paragraph>
<Logcat information>

Steps to reproduce the behavior
1. Start a new post
2. Type something
3. Tap on the Publish button
4. The app crashes
<Text paragraph>

Test Case

```java
public void test() {
    // start a new post
    onView(
        withId(R.id.fab_button))
        .perform(click());

    // type something
    onView(
        withId(R.id.post_title))
        .perform(typeText("GrbcVK9I"));

    // tap on the Publish button
    onView(
        withId(R.id.menu_save_post))
        .perform(click());
}
```
Yakusu Overview

Ontology Extraction

Bug Report Analysis

UI Actions Search

Relevant App

Ontology

Bug Report

Abstract Steps

Test Device

Test Case

UI Actions Search

Test Case
Yakusu Overview

- Ontology Extraction
- Bug Report Analysis
- UI Actions Search

Relevant App

Ontology Extraction

Ontology

Bug Report

Test Case
Yakusu Overview

Ontology Extraction

Bug Report Analysis

UI Actions Search

Ontology

Bug Report

Test Case

Abstract Steps

Test Device
Ontology Extraction

Relevant App

Ontology

UI Element Properties

Example

GrbcVK9I

Content (tap to add text and media)
Ontology Extraction

Relevant App

Ontology

UI Element

Properties

Example

GrbcVK9i

Content (tap to add text and media)
Ontology Extraction

Relevant App → Ontology → UI Element Properties

Example

GrbcVK9l

Content (tap to add text and media)
Bug Report Analysis

Bug Report → Abstract Steps

Dependency Tree (Case #1)

Example → Tap on the Publish button.

Abstract Step

<action, target, properties>

Dependency Tree (Case #2)

Example → Start a new post.

Abstract Step

<action, target, properties>

Ontology

Abstract Steps

Example

Example

Word2Vec

Clause Vector → Cosine Similarity

Property Vector

Start a new post
Bug Report Analysis

Bug Report  Abstract Steps

Dependency Tree (Case #1)

Example Tap on the Publish button.

Abstract Step

\< \text{action} , \text{target} , \text{properties} \>

Dependency Tree (Case #2)

Example Start a new post.

Ontology

\text{word2vec}

\text{Clause Vector} \rightarrow \text{Cosine Similarity} \rightarrow \text{Property Vector}

Test

\< \text{question} , \text{Start a new post} , [] \>
Bug Report Analysis

Bug Report → Abstract Steps

Dependency Tree (Case #1)

Example: Tap on the Publish button.

Abstract Step

\[ \langle \text{action}, \text{target}, \text{properties} \rangle \]

Dependency Tree (Case #2)

Example: Start a new post.

Ontology

word2vec

Clause Vector → Property Vector

Cosine Similarity

\[ \langle ?, \text{Start a new post}, [] \rangle \]
Bug Report Analysis

Bug Report → Abstract Steps

Dependency Tree (Case #1)

Example: Tap on the Publish button.

Abstract Step

Dependency Tree (Case #2)

Example: Start a new post.

Ontology

word2vec

Clause Vector → Cosine Similarity → Property Vector
UI Actions Search

Abstract Steps → Test Case

Abstract Step: {action, target, properties}

UI Action: {UI action, UI element, properties}

Test Device: Relevant App
UI Actions Search

Abstract Steps → Test Case

Abstract Step: <action>, <target>, <properties> → UI Action: <UI action>, <UI element>, <properties>

Test Device

Relevant App
UI Actions Search

Abstract Steps → Test Case

Abstract Step: (action, target, properties) → UI Action: (UI action, UI element, properties)

Test Device

Test Case

relevant App
UI Actions Search

Abstract Step

{Tap, Publish, []}

Find UI Element

word2vec

Publish → UI Elements

Publish, UI element #2, UI element #N

Current Execution

{Tap, Publish, []}

Forked Executions

{Tap, UI element #2, []}

{Tap, UI element #N, []}

Random UI Action Generation
UI Actions Search

Abstract Step

\[
\text{Tap}, \text{ Publish}, \text{ []}
\]

Find UI Element

\[
\text{Publish} \rightarrow \text{word2vec} \rightarrow \text{UI Elements}
\]

UI Elements

\[
\text{Publish}, \text{ UI element } \#2, \ldots, \text{ UI element } \#N
\]

Current Execution

\[
\text{Tap}, \text{ Publish}, \text{ []}
\]

Forked Executions

\[
\text{Tap}, \text{ UI element } \#2, \text{ []}, \ldots, \text{ Tap}, \text{ UI element } \#N, \text{ []}
\]

Random UI Action Generation
UI Actions Search

Abstract Step

\[
\langle \text{Tap}, \text{Publish}, \text{[]} \rangle
\]

Find UI Element

word2vec

Publish

UI Elements

Find UI Element

word2vec

Publish

UI element #2

UI element #N

Current Execution

\[
\langle \text{Tap}, \text{Publish}, \text{[]} \rangle
\]

Forked Executions

\[
\langle \text{Tap}, \text{UI element #2}, \text{[]} \rangle
\]

\[
\langle \text{Tap}, \text{UI element #N}, \text{[]} \rangle
\]

Random UI Action Generation
UI Actions Search

Abstract Step

\[
\langle \text{Tap}, \text{Publish}, [\text{null}] \rangle
\]

Find UI Element

Find UI Element

word2vec

Publish

UI Elements

Publish

UI element #2

UI element #N

Current Execution

\[
\langle \text{Tap}, \text{Publish}, [\text{null}] \rangle
\]

Forked Executions

\[
\langle \text{Tap}, \text{UI element #2}, [\text{null}] \rangle
\]

\[
\langle \text{Tap}, \text{UI element #N}, [\text{null}] \rangle
\]

Random UI Action Generation
UI Actions Search

Abstract Step

{Tap, Publish, []}

Find UI Element

word2vec

Publish → UI Elements

Publish

UI element #2

... 

UI element #N

Current Execution

{Tap, Publish, []}

Forked Executions

{Tap, UI element #2, []}

... 

{Tap, UI element #N, []}

Random UI Action Generation
public void test() {

  // start a new post
  onView(
    withId(R.id.fab_button))
  .perform(click());

  // type something
  onView(
    withId(R.id.post_title))
  .perform(typeText("GrbcVK9I"));

  // tap on the publish button
  onView(
    withId(R.id.menu_save_post))
  .perform(click());
}
Empirical Evaluation

Research Questions:

RQ1: Can YAKUSU translate bug reports written in natural language into executable test cases?

RQ2: What is the cost of running YAKUSU?
Empirical Evaluation

Research Questions:

**RQ1:** Can YAKUSU translate bug reports written in natural language into executable test cases?

**RQ2:** What is the cost of running YAKUSU?
Empirical Evaluation

Research Questions:

RQ1: Can YAKUSU translate bug reports written in natural language into executable test cases?

RQ2: What is the cost of running YAKUSU?
Benchmarks and Setup

Could not Build: x 9
Could not Reproduce Manually: x 29
Available Reports: x 62
Successfully Generated Tests: x 37

GitHub → Bug Report: x 100
Benchmarks and Setup

- Could not Build: x 9
- Could not Reproduce Manually: x 29
- Available Reports: x 62
- Successfully Generated Tests: x 37
Benchmarks and Setup

Could not Build  x 9
Could not Reproduce Manually  x 29
Available Reports  x 62
Successfully Generated Tests  x 37

GitHub  →  Bug Report  x 100
Benchmarks and Setup

Could not Build: x 9
Could not Reproduce Manually: x 29
Available Reports: x 62
Successfully Generated Tests: x 37
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Actions</th>
<th>Steps</th>
<th>Search</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>K-9 MAIL</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>BUTTERKNIFE</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>ODK COLLECT</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>39</td>
<td>PIX-ART MESSENGER</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>YALP STORE</td>
<td>5</td>
<td>-</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>OCRREADER</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>51</td>
<td>WORDPRESS</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>52</td>
<td>SIGNAL</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>57</td>
<td>OPEN EVENT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>68</td>
<td>TAGMO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>69</td>
<td>ANKI DROID</td>
<td>5</td>
<td>-</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>73</td>
<td>K-9 MAIL</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>74</td>
<td>CLUTTR</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECORDER</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
<td>Actions</td>
<td>Steps</td>
<td>Search</td>
<td>Tests</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------</td>
<td>---------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>K-9 MAIL</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>BUTTERKNIFE</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>ODK COLLECT</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>39</td>
<td>PIX-ART MESSENGER</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>YALP STORE</td>
<td>5</td>
<td>-</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>OCRREADER</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>51</td>
<td>WORDPRESS</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>52</td>
<td>SIGNAL</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>57</td>
<td>OPEN EVENT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>68</td>
<td>TAGMO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>69</td>
<td>ANKI DROID</td>
<td>5</td>
<td>-</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>73</td>
<td>K-9 MAIL</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>74</td>
<td>CLUTTR</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECORDE</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
<td>Actions</td>
<td>Steps</td>
<td>Search</td>
<td>Tests</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------</td>
<td>---------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>K-9 MAIL</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>BUTTERKNIFE</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>ODK COLLECT</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>39</td>
<td>PIX-ART MESSENGER</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>YALP STORE</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>50</td>
<td>OCRREADER</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>52</td>
<td>SIGNAL</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>57</td>
<td>OPEN EVENT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>68</td>
<td>TAGMO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>69</td>
<td>ANKI DROID</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>73</td>
<td>K-9 MAIL</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>74</td>
<td>CLUTTR</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECORDEr</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ID</td>
<td>Name</td>
<td>Actions</td>
<td>Steps</td>
<td>Search</td>
<td>Tests</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------</td>
<td>---------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>08</td>
<td>RedReader</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>K-9 Mail</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>BUTTERKNIFE</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>ODK COLLECT</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>39</td>
<td>PIX-ART MESSENGER</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>YALP STORE</td>
<td>5</td>
<td>-</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>OCRReader</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>51</td>
<td>WORDPRESS</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>52</td>
<td>SIGNAL</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>57</td>
<td>OPEN EVENT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>68</td>
<td>TAGMO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>69</td>
<td>ANKI DROID</td>
<td>5</td>
<td>-</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>73</td>
<td>K-9 Mail</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>74</td>
<td>Cluttr</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>84</td>
<td>K-9 Mail</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECORDER</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
9 bug reports had at least one implicit action
7 bug reports had at least one step without explicit action
19 bug reports reproduced exploring only one execution
6 bug reports reproduced exploring more than one execution
4 test cases with additional actions
### RQ1

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Actions</th>
<th>Steps</th>
<th>Search</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

**9 bug reports had at least one implicit action**

7 bug reports had at least one step without explicit action

19 bug reports reproduced exploring only one execution

6 bug reports reproduced exploring more than one execution

4 test cases with additional actions

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Actions</th>
<th>Steps</th>
<th>Search</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECORDER</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>3</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
### RQ1

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Actions</th>
<th>Steps</th>
<th>Search</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

**9 bug reports had at least one implicit action**

**7 bug reports had at least one step without explicit action**

**19 bug reports reproduced exploring only one execution**

**6 bug reports reproduced exploring more than one execution**

**4 test cases with additional actions**
## RQ1

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Actions</th>
<th>Steps</th>
<th>Search</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

9 bug reports had at least one implicit action

7 bug reports had at least one step without explicit action

19 bug reports reproduced exploring only one execution

6 bug reports reproduced exploring more than one execution

4 test cases with additional actions
### RQ1

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Actions</th>
<th>Steps</th>
<th>Search</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

- **9** bug reports had at least one implicit action
- **7** bug reports had at least one step without explicit action
- **19** bug reports reproduced exploring only one execution
- **6** bug reports reproduced exploring more than one execution
- **4** test cases with additional actions
## RQ1

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Actions</th>
<th>Steps</th>
<th>Search</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$A_e$</td>
<td>$A_i$</td>
<td>$AS$</td>
<td>$AS_g$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

9 bug reports had at least one implicit action

7 bug reports had at least one step without explicit action

19 bug reports reproduced exploring only one execution

6 bug reports reproduced exploring more than one execution

4 test cases with additional actions
RQ1: Can YAKUSU translate bug reports written in natural language into executable test cases?

We consider the results encouraging as YAKUSU is already able to generate test cases for a good number of bug reports.
RQ2

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>$T_{oe}$</th>
<th>$T_{bra}$</th>
<th>$T_{uias}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>54s</td>
<td>18s</td>
<td>1m12s</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2m34s</td>
<td>15s</td>
<td>1m07s</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1m23s</td>
<td>17s</td>
<td>6m57s</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>25s</td>
<td>19s</td>
<td>1m25s</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>43s</td>
<td>15s</td>
<td>5m08s</td>
</tr>
<tr>
<td>23</td>
<td>K-9 MAIL</td>
<td>32s</td>
<td>16s</td>
<td>1m00s</td>
</tr>
<tr>
<td>25</td>
<td>NEXTCLOUD</td>
<td>34s</td>
<td>16s</td>
<td>2m08s</td>
</tr>
<tr>
<td>27</td>
<td>BUTTERKNIFE</td>
<td>13s</td>
<td>16s</td>
<td>09s</td>
</tr>
<tr>
<td>35</td>
<td>ODK COLLECT</td>
<td>30s</td>
<td>18s</td>
<td>3m51s</td>
</tr>
<tr>
<td>39</td>
<td>PIX-ART MESSENGER</td>
<td>43s</td>
<td>15s</td>
<td>31s</td>
</tr>
<tr>
<td>40</td>
<td>YALP STORE</td>
<td>16s</td>
<td>17s</td>
<td>2m25s</td>
</tr>
<tr>
<td>50</td>
<td>OCRREADER</td>
<td>22s</td>
<td>17s</td>
<td>1m33s</td>
</tr>
<tr>
<td>51</td>
<td>WORDPRESS</td>
<td>1m35s</td>
<td>16s</td>
<td>3m18s</td>
</tr>
<tr>
<td>52</td>
<td>SIGNAL</td>
<td>1m16s</td>
<td>17s</td>
<td>18m10s</td>
</tr>
<tr>
<td>57</td>
<td>OPEN EVENT</td>
<td>36s</td>
<td>20s</td>
<td>2m48s</td>
</tr>
<tr>
<td>68</td>
<td>TAGMO</td>
<td>17s</td>
<td>16s</td>
<td>31s</td>
</tr>
<tr>
<td>69</td>
<td>ANKI DROID</td>
<td>29s</td>
<td>17s</td>
<td>28m55s</td>
</tr>
<tr>
<td>73</td>
<td>K-9 MAIL</td>
<td>35s</td>
<td>18s</td>
<td>3m22s</td>
</tr>
<tr>
<td>74</td>
<td>CLUTTR</td>
<td>26s</td>
<td>15s</td>
<td>24s</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>32s</td>
<td>15s</td>
<td>2m20s</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>32s</td>
<td>17s</td>
<td>45s</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>39s</td>
<td>18s</td>
<td>1m25s</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECORDER</td>
<td>15s</td>
<td>17s</td>
<td>54s</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>37s</td>
<td>23s</td>
<td>9m28s</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>15s</td>
<td>18s</td>
<td>58s</td>
</tr>
</tbody>
</table>
### RQ2

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$T_{oe}$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>54s</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2m34s</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1m23s</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>25s</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>42s</td>
</tr>
<tr>
<td>73</td>
<td>K-9 MAIL</td>
<td>35s</td>
</tr>
<tr>
<td>74</td>
<td>CLUTTR</td>
<td>26s</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>32s</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>32s</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>39s</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECODER</td>
<td>15s</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>37s</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>15s</td>
</tr>
</tbody>
</table>

UI Actions search phase takes the longest on average

Average analysis time is 5m00s and median time is 2m58s

Longest analysis time is 29m41s
## RQ2

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Toe</th>
<th>Bra</th>
<th>UIas</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>54s</td>
<td>18s</td>
<td>1m12s</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2m34s</td>
<td>15s</td>
<td>1m07s</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1m23s</td>
<td>17s</td>
<td>6m57s</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>25s</td>
<td>19s</td>
<td>1m25s</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>43s</td>
<td>15s</td>
<td>5m08s</td>
</tr>
</tbody>
</table>

UI Actions search phase takes the longest on average

Average analysis time is 5m00s and median time is 2m58s

Longest analysis time is 29m41s
## RQ2

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>$T_{oe}$</th>
<th>$T_{bra}$</th>
<th>$T_{uias}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>54s</td>
<td>18s</td>
<td>1m12s</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2m34s</td>
<td>15s</td>
<td>1m07s</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1m23s</td>
<td>17s</td>
<td>6m57s</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>25s</td>
<td>19s</td>
<td>1m25s</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>42s</td>
<td>15s</td>
<td>5m08s</td>
</tr>
<tr>
<td>73</td>
<td>K-9 MAIL</td>
<td>35s</td>
<td>18s</td>
<td>3m22s</td>
</tr>
<tr>
<td>74</td>
<td>CLUTTR</td>
<td>26s</td>
<td>15s</td>
<td>24s</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>32s</td>
<td>15s</td>
<td>2m20s</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>32s</td>
<td>17s</td>
<td>45s</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>39s</td>
<td>18s</td>
<td>1m25s</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECORDER</td>
<td>15s</td>
<td>17s</td>
<td>54s</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>37s</td>
<td>23s</td>
<td>9m28s</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>15s</td>
<td>18s</td>
<td>58s</td>
</tr>
</tbody>
</table>

UI Actions search phase takes the longest on average.

Average analysis time is 5m00s and median time is 2m58s.

Longest analysis time is 29m41s.
### RQ2

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>$T_{oe}$</th>
<th>$T_{bra}$</th>
<th>$T_{uias}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>54s</td>
<td>18s</td>
<td>1m12s</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2m34s</td>
<td>15s</td>
<td>1m07s</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1m23s</td>
<td>17s</td>
<td>6m57s</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>25s</td>
<td>19s</td>
<td>1m25s</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>43s</td>
<td>15s</td>
<td>5m28s</td>
</tr>
<tr>
<td>73</td>
<td>K-9 MAIL</td>
<td>35s</td>
<td>18s</td>
<td>3m22s</td>
</tr>
<tr>
<td>74</td>
<td>CLUTTR</td>
<td>26s</td>
<td>15s</td>
<td>24s</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>32s</td>
<td>15s</td>
<td>2m20s</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>32s</td>
<td>17s</td>
<td>45s</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>39s</td>
<td>18s</td>
<td>1m25s</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECODER</td>
<td>15s</td>
<td>17s</td>
<td>54s</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>37s</td>
<td>23s</td>
<td>9m28s</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>15s</td>
<td>18s</td>
<td>58s</td>
</tr>
</tbody>
</table>

UI Actions search phase takes the longest on average. Average analysis time is 5m00s and median time is 2m58s. Longest analysis time is 29m41s.
RQ2

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$T_{oe}$</td>
</tr>
<tr>
<td>01</td>
<td>TACHIYOMI</td>
<td>54s</td>
</tr>
<tr>
<td>03</td>
<td>TWIDERE</td>
<td>2m34s</td>
</tr>
<tr>
<td>05</td>
<td>SIGNAL</td>
<td>1m23s</td>
</tr>
<tr>
<td>08</td>
<td>REDREADER</td>
<td>25s</td>
</tr>
<tr>
<td>14</td>
<td>SILENCE</td>
<td>42s</td>
</tr>
<tr>
<td>73</td>
<td>K-9 MAIL</td>
<td>35s</td>
</tr>
<tr>
<td>74</td>
<td>CLUTTR</td>
<td>26s</td>
</tr>
<tr>
<td>78</td>
<td>NEXTCLOUD</td>
<td>32s</td>
</tr>
<tr>
<td>84</td>
<td>K-9 MAIL</td>
<td>32s</td>
</tr>
<tr>
<td>95</td>
<td>MIFOSX</td>
<td>39s</td>
</tr>
<tr>
<td>96</td>
<td>SCREENRECORDER</td>
<td>15s</td>
</tr>
<tr>
<td>97</td>
<td>NEXTCLOUD</td>
<td>37s</td>
</tr>
<tr>
<td>99</td>
<td>FLASHCARDS</td>
<td>15s</td>
</tr>
</tbody>
</table>

RQ2: What is the cost of running YAKUSU?

The cost of running YAKUSU is fairly low and YAKUSU can be used to monitor bug reports and generate test cases throughout the day as opposed to overnight only.
Future Work

- Extend ontology
- Learn macro-step associated with the app
- Interpret non-crashing failures from bug reports
- Generate test cases from written/spoken test specifications
- Identify duplicate bug reports
Future Work

- Extend ontology
- Learn macro-step associated with the app
- Interpret non-crashing failures from bug reports
- Generate test cases from written/spoken test specifications
- Identify duplicate bug reports
Future Work

- Extend ontology
- Learn macro-step associated with the app
- Interpret non-crashing failures from bug reports
- Generate test cases from written/spoken test specifications
- Identify duplicate bug reports
Future Work

- Extend ontology
- Learn macro-step associated with the app
- Interpret non-crashing failures from bug reports
- Generate test cases from written/spoken test specifications
- Identify duplicate bug reports
Future Work

- Extend ontology
- Learn macro-step associated with the app
- Interpret non-crashing failures from bug reports
- Generate test cases from written/spoken test specifications
- Identify duplicate bug reports
Future Work

- Extend ontology
- Learn macro-step associated with the app
- Interpret non-crashing failures from bug reports
- Generate test cases from written/spoken test specifications
- Identify duplicate bug reports
Opportunities at Georgia Tech

Research Area

Software testing and program analysis

Openings

Postdocs, internships, PhD positions

Alessandro Orso

Email: orso@cc.gatech.edu
Summary
Summary
Summary
Summary