Automatic Debugging of Android Applications

Pedro Machado

Supervisors:
Rui Maranhão (PhD)
José Campos (MSc)
“Context”
Context
Context

Apple

Android

Windows Phone

BlackBerry

Symbian
Context

Source: Gartner
Software Development Process

- Design
- Implement
- Test
- Release
- Debug
“Challenges”
Debugging

Money

Time
Challenges

- Debugging is a manual task
- Resource constraints
- Very specific architectures
Challenges

Are mobile apps...

... reliable?
Hypothesis

Static and dynamic analysis improve the reliability of Mobile Apps.
“Related Work”
Android Debugging tools [6]
Fault localization tools

Gzoltar [4]  

Tarantula [5]
“Motivational Example”
Spectrum-based fault localization

class CharCount  {
    static void count(String s)
    {
        int let, dig, other;
        for(int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if ('A'<=c && 'Z'>=c) let += 2; /* FAULT */
            else if ('a'<=c && 'z'>=c) let += 1;
            else if ('0'<=c && '9'>=c) dig += 1;
            else if (isprint(c)) other += 1; }
        System.out.println(let + " " + dig + " " + other);
    }
}

Error Vector
Lint static analysis

Lint yields issues that indicate potential bugs.

Issues are characterized by their:

- **Category**
- **Severity**
  - Warning
  - Error
  - Fatal
- **Priority** (values from 1 to 10);
Motivational Example

String:
abc123#A
Result:
let: 5 dig: 3 other: 1

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="example.charactercount"
    android:versionName="1.0">
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17"/>
    <application
        android:allowBackup="true"
        android:icon="/drawable/ic_launcher"
        android:label="/string/app_name"
        android:theme="/style/AppTheme">
        <!-- Without this activity the program will fail
                <activity
                    android:name="example.charactercount.CharacterCountActivity"
                    android:label="/string/app_name">
                    <intent-filter>
                        <action android:name="android.intent.action.MAIN"/>
                        <category android:name="android.intent.category.LAUNCHER"/>
                    </intent-filter>
                </activity> -->
    </application>
</manifest>
```
“Tool”
Visualizations

workspace. project. packageroot. package. file. class. method. line
Static and Dynamic Analysis

\[ L_c(m) = \min \left( \frac{\bar{I}_P(m)}{\max Pr}, \bar{I}_S(m), 1 \right) \]

\[ C(m) = \min \left( s_O(m) \cdot (1 + L_c(m)), 1 \right) \]
Static and Dynamic Analysis

Static

Dynamic + Static
MZoltar's Workflow

Instrument

ASM | ASMDex | JaCoCo

Run Tests

ADT / ADB

Obtaining program spectra

Logcat | Sockets | Files | EMMA
“Evaluation”
Research Questions

- Is the instrumentation overhead negligible?
- Do diagnostics remain accurate under Android’s constrained environment?
- Does Lint integration contribute to a better diagnostic quality?
## Experimental Setup

<table>
<thead>
<tr>
<th>Subject</th>
<th>LOC</th>
<th>Tests</th>
<th>Coverage</th>
<th>Resources LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharCount</td>
<td>148</td>
<td>10</td>
<td>92.2%</td>
<td>115</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>32911</td>
<td>14</td>
<td>0.7%</td>
<td>7673</td>
</tr>
<tr>
<td>Google Authenticator</td>
<td>3659</td>
<td>170</td>
<td>76.6%</td>
<td>5275</td>
</tr>
<tr>
<td>StarDroid</td>
<td>13783</td>
<td>187</td>
<td>29.7%</td>
<td>2694</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>Subject</th>
<th>Original</th>
<th>Instrumented</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharCount</td>
<td>1.82s</td>
<td>1.86s</td>
<td>2%</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>1.25s</td>
<td>1.35s</td>
<td>8%</td>
</tr>
<tr>
<td>Google Authenticator</td>
<td>80.49s</td>
<td>87.26s</td>
<td>8%</td>
</tr>
<tr>
<td>StarDroid</td>
<td>14.70s</td>
<td>15.46s</td>
<td>5%</td>
</tr>
</tbody>
</table>

Average Overhead: **5.75%**
Metrics – Diagnostic Quality

- Average number of lines to inspect
- Coverage density
Multiple Faults / Run

Charcount

Google Authenticator

Connectbot

Stardroid
Multiple Tests / Run

Charcount

Connectbot

Google Authenticator

Stardroid
Multiple Tests / Run

Average Time Reduction: \textbf{79\%}

Average $C_d$ Growth : \textbf{74\%}
## Lint analysis results

<table>
<thead>
<tr>
<th>Subject</th>
<th>Dynamic</th>
<th>Dynamic + Static</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bug 1</td>
<td>Bug 2</td>
</tr>
<tr>
<td>CharCount</td>
<td>148</td>
<td>148</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>32911</td>
<td>32911</td>
</tr>
<tr>
<td>Google Authenticator</td>
<td>3659</td>
<td>3659</td>
</tr>
<tr>
<td>StarDroid</td>
<td>13783</td>
<td>13783</td>
</tr>
</tbody>
</table>

Average $C_d$ Reduction: **99.9%**
“Conclusions”
Conclusions

- The developer only has to inspect an average of 5 out of an average of 12625 lines, before finding the bug;

- Negligible instrumentation overhead (5.75%), despite the Android constrained environment;

- Grouping tests cases reduces time by 79%, but decreases diagnostic quality by 74%;

- Lint integration reduces the number of lines to inspect by 99.9%.
Conclusions

- Reduce development costs
- Reduce time to market
- Increase quality and reliability
Future Work

- Port MZoltar to the Android Studio IDE;
- Port MZoltar to other mobile technologies;
- Correlate Lint issues and Runtime failures;
- Perform an User Study to evaluate MZoltar;
- Investigate a bug prediction approach.
Publications

Published:

• Pedro Machado, José Campos and Rui Abreu, *Automatic Debugging of Android Applications*. In IJUP’12, Porto, Portugal

• Pedro Machado, José Campos and Rui Abreu, *MZoltar: Automatic Debugging of Android Applications*. In DeMobile’13, St Petesburg, Russia. (Co-located with ESEC/FSE)

Submitted:

• Pedro Machado, José Campos and Rui Abreu. *Combining Static and Dynamic Analysis in Mobile Apps Fault Localization*. In ISSRE’13, Pasadena, CA, USA
“Questions, anyone?”
## Work Plan – 2nd Semester

<table>
<thead>
<tr>
<th>Months</th>
<th>February</th>
<th>March</th>
<th>April</th>
<th>May</th>
<th>June</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities/Weeks</td>
<td>11</td>
<td>18</td>
<td>25</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>Framework</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture particularities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration with Gzoltar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specific visualizations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing and validating results</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Article writing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thesis writing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Motivation

Global smart phone sales vs. PC sales, 2011 (according to Canalys)

<table>
<thead>
<tr>
<th>Device</th>
<th>Shipments 2011 (millions)</th>
<th>Annual growth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Smartphones</strong></td>
<td>487.7</td>
<td><strong>62.7%</strong></td>
</tr>
<tr>
<td><strong>Tablets</strong></td>
<td>63.2</td>
<td><strong>274.2%</strong></td>
</tr>
<tr>
<td>Notebooks</td>
<td>209.6</td>
<td>7.5%</td>
</tr>
<tr>
<td>Desktops</td>
<td>112.4</td>
<td>2.3%</td>
</tr>
<tr>
<td>Netbooks</td>
<td>29.4</td>
<td>-25.3%</td>
</tr>
</tbody>
</table>

Source: [Canalys (Feb 2012)](https://www.canalys.com)

## Work Plan – 1st Semester

<table>
<thead>
<tr>
<th>Months</th>
<th>October</th>
<th>November</th>
<th>December</th>
<th>January</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities\Weeks</td>
<td>8 15 22 29</td>
<td>5 12 19 26</td>
<td>3 10 17 24 31</td>
<td>7 14 21 28</td>
</tr>
<tr>
<td>SFL Concept</td>
<td>🔴</td>
<td>🔴</td>
<td>🔴</td>
<td></td>
</tr>
<tr>
<td>Android Testing</td>
<td>🔴</td>
<td>🔴</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture particularities</td>
<td>🔴</td>
<td>🔴</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gzoltar</td>
<td>🔴</td>
<td>🔴</td>
<td>🔴</td>
<td></td>
</tr>
<tr>
<td>Preparation of first presentation</td>
<td>🔴</td>
<td>🔴</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADT</td>
<td></td>
<td></td>
<td>🔴</td>
<td>🔴</td>
</tr>
<tr>
<td>Preparation of second presentation</td>
<td></td>
<td></td>
<td>🔴</td>
<td>🔴</td>
</tr>
<tr>
<td>Writing the state of the art</td>
<td></td>
<td></td>
<td>🔴</td>
<td>🔴</td>
</tr>
</tbody>
</table>
Metrics – Diagnostic Quality

\[ \theta = |\{j|s_o(m) > s_o(d_*)\}|, \quad 1 \leq j \leq M \]

\[ \phi = |\{j|s_o(m) \geq s_o(d_*)\}|, \quad 1 \leq j \leq M \]

\[ C_d = \frac{\theta + \phi - 1}{2} \]

Diagnostic Quality

Density

\[ \bar{\rho} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} a_{ij}}{N \cdot M} \]
Similarity Coefficients

\[ s_J(j) = \frac{a_{11}(j)}{a_{11}(j) + a_{01}(j) + a_{10}(j)} \]

\[ s_T(j) = \frac{a_{11}(j)}{a_{11}(j) + a_{01}(j)} \frac{a_{11}(j) + a_{01}(j)}{a_{11}(j) + a_{01}(j) + a_{10}(j)} + \frac{a_{10}(j)}{a_{10}(j) + a_{00}(j)} \]

\[ s_O(j) = \frac{a_{11}(j)}{\sqrt{(a_{11}(j) + a_{01}(j)) \cdot (a_{11}(j) + a_{10}(j))}} \]


