Resource Interaction Failures in Mobile Applications: A Challenge for the SPL Community and Other Perspectives

Euler Marinho
Summary

- An overview of the last submitted paper
- Other perspectives
  - Current research
AN OVERVIEW OF THE LAST SUBMITTED (SHORT) PAPER
Introduction

- Mobile devices have a rich set of resources
- "Resource" refers to sensors, radios, and user-controlled options
- User interaction with devices can enable or disable the resources
- Unexpected application behavior can occur in specific resource settings
- However, the testing of all input combinations is impracticable
Sampling Strategies

- Resource interactions are like Feature interactions
- Resource settings are 14-tuple of resource and state pairs
- Sampling strategies are alternatives for decreasing the testing effort
- Random (30), One Enabled (14), One Disabled (14), Most Enabled Disabled (2), Pairwise (8)
Proposed Challenge

- SPLC participants must propose testing strategies for mobile applications
  - Taking resource interactions into account
- The failure detection capability and the effectiveness must be higher than our baseline
  - Increase the number of unique detected failures and minimize the number of tested settings
- Solution efficiency (SE) \[ SE = \frac{FailingSettings}{TotalSettings} \]
Dataset

- 20 Android applications
- 14 target resources
  - Auto Rotate, Battery Saver, Bluetooth, Camera, Do Not Disturb, Location, Mobile Data, Wi-Fi, Accelerometer, Gyroscope, Light, Magnetometer, Orientation, Proximity
- Extended test suites
Evaluation metrics

- **Size metrics**
  - LOC
  - Number of packages, classes, methods

- **Test suite metrics**
  - LOC
  - Number of test cases

- **Declared resources**
Dataset Excerpt

<table>
<thead>
<tr>
<th>NAME</th>
<th>LOC</th>
<th>#Test Cases</th>
<th>Test LOC</th>
<th>RESOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnkiDroid</td>
<td>158 K</td>
<td>164</td>
<td>2,770</td>
<td>Cam, MD, Wi-Fi</td>
</tr>
<tr>
<td>CovidNow</td>
<td>2 K</td>
<td>21</td>
<td>540</td>
<td>MD, Wi-Fi</td>
</tr>
<tr>
<td>Iosched</td>
<td>27 K</td>
<td>9</td>
<td>473</td>
<td>Loc, MD, Wi-Fi</td>
</tr>
<tr>
<td>Mixin-Messenger</td>
<td>168 K</td>
<td>160</td>
<td>3,732</td>
<td>BT, Cam, Loc, MD, Wi-Fi</td>
</tr>
<tr>
<td>Moonshot</td>
<td>455 K</td>
<td>28</td>
<td>464</td>
<td>MD, Wi-Fi</td>
</tr>
<tr>
<td>Radio-Droid</td>
<td>22 K</td>
<td>23</td>
<td>1,735</td>
<td>BT, MD, Wi-Fi</td>
</tr>
<tr>
<td>WordPress</td>
<td>347 K</td>
<td>115</td>
<td>3,674</td>
<td>Cam, MD, Wi-Fi</td>
</tr>
</tbody>
</table>

BT - Bluetooth
Cam - Camera
Loc - Location
MD - Mobile Data
Test suite instrumentation

- Functional tests are the target
  - Android APIs for interacting with the device
- Extension by means of UI Automator
- Each test class is extended with instrumentation code
- Before each test case the instrumentation code is executed
- Test reports are processed
## Test Reports

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILING SETTINGS</th>
<th>SOLUTION EFFICIENCY</th>
<th>#FAILURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CovidNow</td>
<td>32</td>
<td>0.47</td>
<td>2</td>
</tr>
<tr>
<td>Lockwise</td>
<td>68</td>
<td>1.00</td>
<td>4</td>
</tr>
<tr>
<td>Mixin-Messenger</td>
<td>20</td>
<td>0.29</td>
<td>2</td>
</tr>
<tr>
<td>Nl-covid19</td>
<td>55</td>
<td>0.81</td>
<td>6</td>
</tr>
<tr>
<td>OwnTracks</td>
<td>68</td>
<td>1.00</td>
<td>3</td>
</tr>
<tr>
<td>PocketHub</td>
<td>4</td>
<td>0.06</td>
<td>1</td>
</tr>
<tr>
<td>SpaceXFollower</td>
<td>68</td>
<td>1.00</td>
<td>4</td>
</tr>
<tr>
<td>Threema</td>
<td>33</td>
<td>0.48</td>
<td>1</td>
</tr>
<tr>
<td>Vocable</td>
<td>24</td>
<td>0.35</td>
<td>7</td>
</tr>
<tr>
<td>WordPress</td>
<td>37</td>
<td>0.54</td>
<td>11</td>
</tr>
</tbody>
</table>
Example of Use

- Settings are provided in CSV files with only enabled resources

  - Location, Bluetooth, Battery_Saver, Do_Not_Disturb, Accelerometer, Light
Example of Output

- Vocable is a communication tool for individuals who are speech impaired
- It uses the ARCore SDK to track the user's head movements
  - To understand where the user is looking on the screen
- When both Mobile Data and Wi-Fi are disabled (*verifyDefaultTextAppears* test)
  - ARCore fatal exception
CURRENT RESEARCH
Resource Interaction Faults

- Characterize faults behind the failures
  - How to identify faulty classes?
    - Failures are related to the test framework scope
    - Android event-driven nature is a challenge for debugging activities
- Spectrum based Fault Localization
- Extension of our SBES 2023 paper
Resource Interaction Faults

- Compare faults generated by traditional mutation operators and manually injected faults
- Faults injected using Bug Fix Patterns (Pan et al. 2009)
  - Using the five most common Bug Fix Patterns (Campos and Maia 2017)
Bug fix patterns (Pan et al. 2009)

- Change of IF Condition Expression (IF-CC):
  
  - `if (listBox.getSelectedIndex() == 0)`
  + `if (listBox.getSelectedIndex() > 0)`

- Method Call with different actual parameter values (MC-DAP)
  
  - `String.getBytes("UTF-8");`
  + `String.getBytes("ISO-8859-1");`
Bug fix patterns – Parte 2

- Method Call with different number of parameters or different types of parameters (MC-DNP)

```java
- getsolrquery(f.getfilter());
+ getsolrquery(f.getfilter(), analyzer);
```

- Change of Assignment Expression (AS-CE)

```java
- names[0] = person.getName();
+ names[0] = employees[0].getName();
```
Bug fix patterns – Parte 3

- Addition of IF Precondition Check (IF-APC)

```java
- repo.getFileContent(path);
+ if (repo != null && path != null)
+   repo.getFileContent(path);
```
Questions?