Characterizing Failures in Mobile OSes: A Case Study with Android and Symbian

Amiya Kumar Maji, Kangli Hao, Salmin Sultana, Saurabh Bagchi
Dependable Computing Systems Lab (DCSL)
School of Electrical and Computer Engineering
Purdue University

Emergence of Smartphones

• 14% of 1.2 billion mobile phone sales in 2009 are smartphones (Gartner)
• 19% of 1.6 billion mobile phone sales in 2010 are smartphones (Gartner)
  – 72.1% increase compared to 2009
• 25% of mobile phone sales in Q2 2011 are smartphones (Gartner)
• Smartphones expected to be the majority in US mobile market by end of 2011 (Nielsen)
The Changing Face of Mobile OSes

• “There should be nothing that users can access on their desktop that they can’t access on their cell phone.”
  – Andy Rubin

• Open source initiatives by Android and Symbian
• Public forums for bug reporting and bug fixes
How Reliable are Smartphones?

---

**Warranty Claims**
- iPhone 2.1%
- Motorola Droid 2.3%
- HTC 3.7%
- BlackBerry 6.3%

---

- Earlier study by Cinque et al. [DSN’07] looks at failure of Symbian phones using failure event logger

---

**Our Objectives**

- To determine failure characteristics of smartphones from public bug databases
- Part I:
  - How failures manifest?
  - Are failures in Android and Symbian comparable?
- Part II:
  - Bug fix analysis
  - Tension between customizability, complexity, and bug density
Part I
Manifestation of Failures

Overview of Android
Overview of Symbian

Data Collection

- **Source:**
  - Android Issue Reports:
    - Posted by app developers or users (with sufficient details)
      
      http://code.google.com/p/android/issues/
  - Symbian Bug Tracker:
    - Posted primarily by developers
      
      http://developer.symbian.org/bugs/
An Example Bug Report in Android

Dataset Summary

- Selection keywords:
  - Crash, shutdown, freeze, broken, failure, error, exception, and security
- Further data pruning due to:
  - Duplicates, pre-release bugs, too little details
  - Questions, enhancements
- Android
  - Timespan: October 2008-October 2009
  - Number of bug reports: 628
- Symbian
  - Timespan: Feb 2010-April 2010
  - Number of bug reports: 153
Location of Manifestation of Faults

- Initial counts of faulty applications/libraries
  - Android: 55
  - Symbian: 41

- Aggregate related packages into “segments”
  - Eclipse, Android Dev Tool (ADT), Android Debug Bridge (ADB) as Development Tools
  - wrtttools, web, websrv, and webuis as Web

- Count of segments
  - Android: 18
  - Symbian: 15

Distribution of Bugs: Android

- Count indicates unique failures
- Failure of Dev tools, Doc-install is a concern for app development
- Failure of Web browser, Multimedia degrade user experience
Distribution of Bugs: Symbian

- Lots of build failures
- Codebase not yet stable

Comparing the Graphs

- 4 of top 6 failure-prone segments are identical
  - Web, Multimedia, Development Tools, Documentation and Installation
- Less bugs in Kernel and Drivers
- Failure of Development Tools is a concern
- Persistence of bugs
  - More than 90% are permanent in nature (can be reproduced predictably)
Looking at User Forums

- T-Mobile G1 (Android) User Forum
  - 105 failure reports related to Messaging, Google Applications, Phone and Data Connections, Operating System and Software Development
  
  - Most frequent failures
    - Mail Client (15)
    - SD Card (11)
    - Media Player (9)
    - Messaging (9)
    - GPS and Location (8)
    - Web Browser (8)
  
  - Recovery actions similar to Cinque et al. [DSN’07]
    - Restart application, wait for some time, restart phone, modify settings, take out battery, factory reset, update firmware etc.

Part II
Analysis of Bug Fixes
Data Collection

- **Source:**
  - Android Code Review:
    
    https://review.source.android.com

- **Timespan:** October 2008-October 2009

- **Count:** 233 bug fixes from 29 projects

- **Example**

  ```java
  try {
    display = labels[type - 1];
  } catch (ArrayIndexOutOfBoundsException e) {
    display = labels[PeoplePhones.TYPE_HOME - 1];
  }
  ```

  Old Version

  ```java
  else {
    display = labels[Organizations.TYPE_WORK - 1];
  }
  ```

  New Version

Categorization of Code Modifications

- Classify programmer errors responsible for failure

- **Categories:**
  - Add/modify attr value
  - Add/modify cond
  - Modify settings
  - Add/modify func call
  - Lock problems
  - Add/modify lib ref etc.
Categories for Bug Fixes

- 77% minor code change
- 23% major change

Observations

- Android is relatively new and still undergoing major modifications
- Detailed specification of program behavior can avoid significant number of bugs (specially in add/modify cond)
  - if statement missing else clause
- Modify settings is third largest category in bug fixes
  - Customizability does have its negative impact!
Analysis of Environment Variables

<table>
<thead>
<tr>
<th></th>
<th># env vars</th>
<th>Total refs</th>
<th>Max refs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android 1.1</td>
<td>62</td>
<td>819</td>
<td>577</td>
</tr>
<tr>
<td>Android 1.5</td>
<td>63</td>
<td>854</td>
<td>584</td>
</tr>
<tr>
<td>Android 1.6</td>
<td>76</td>
<td>1545</td>
<td>584</td>
</tr>
<tr>
<td>Android 2.0</td>
<td>82</td>
<td>2083</td>
<td>592</td>
</tr>
<tr>
<td>Linux Kernel 2.6.32</td>
<td>127</td>
<td>953</td>
<td>158</td>
</tr>
</tbody>
</table>

- Number of environment variables steadily increasing in Android

Distribution of References to Environment Variables

- Android: Majority of references to only a few env variables
### Android: Cyclomatic Complexity vs. Bug Density

<table>
<thead>
<tr>
<th>Projects</th>
<th>Bug Density $\times 10^4$</th>
<th># Bugs</th>
<th>SLOC</th>
<th>Avg. Cyclomatic</th>
<th>Max. Cyclomatic</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel/omap</td>
<td>0.04</td>
<td>21</td>
<td>5,311,427</td>
<td>1.12</td>
<td>4973</td>
</tr>
<tr>
<td>kernel/msm</td>
<td>0.06</td>
<td>29</td>
<td>4,724,260</td>
<td>5.60</td>
<td>4973</td>
</tr>
<tr>
<td>kernel/common</td>
<td>0.07</td>
<td>31</td>
<td>4,688,175</td>
<td>5.82</td>
<td>4973</td>
</tr>
<tr>
<td>dalvik</td>
<td>0.18</td>
<td>14</td>
<td>771,865</td>
<td>2.23</td>
<td>766</td>
</tr>
<tr>
<td>development</td>
<td>0.46</td>
<td>10</td>
<td>216,344</td>
<td>2.18</td>
<td>169</td>
</tr>
<tr>
<td>framework/base</td>
<td>0.79</td>
<td>51</td>
<td>645,978</td>
<td>2.40</td>
<td>221</td>
</tr>
<tr>
<td>packages/apps/camera</td>
<td>1.33</td>
<td>2</td>
<td>14,962</td>
<td>2.15</td>
<td>20</td>
</tr>
<tr>
<td>packages/apps/mms</td>
<td>1.74</td>
<td>4</td>
<td>23,013</td>
<td>2.02</td>
<td>46</td>
</tr>
<tr>
<td>system/core</td>
<td>1.90</td>
<td>13</td>
<td>68,798</td>
<td>4.31</td>
<td>167</td>
</tr>
<tr>
<td>hardware/msm7k</td>
<td>2.42</td>
<td>3</td>
<td>12,382</td>
<td>4.00</td>
<td>23</td>
</tr>
</tbody>
</table>

### Symbian: Cyclomatic Complexity vs. Bug Density

<table>
<thead>
<tr>
<th>Segments</th>
<th>Bug Density $\times 10^4$</th>
<th># Bugs</th>
<th>SLOC</th>
<th>Avg. Cyclomatic</th>
<th>Max. Cyclomatic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel and OS Services</td>
<td>0.03</td>
<td>12</td>
<td>3,684,192</td>
<td>3.02</td>
<td>1470</td>
</tr>
<tr>
<td>Security</td>
<td>0.08</td>
<td>6</td>
<td>752,148</td>
<td>2.29</td>
<td>134</td>
</tr>
<tr>
<td>Multimedia</td>
<td>0.12</td>
<td>22</td>
<td>1,866,577</td>
<td>2.44</td>
<td>558</td>
</tr>
<tr>
<td>Web</td>
<td>0.17</td>
<td>31</td>
<td>1,807,828</td>
<td>3.01</td>
<td>2442</td>
</tr>
<tr>
<td>HomeScreen</td>
<td>0.38</td>
<td>10</td>
<td>263,305</td>
<td>2.25</td>
<td>149</td>
</tr>
<tr>
<td>Build Pkg</td>
<td>0.63</td>
<td>19</td>
<td>299,868</td>
<td>2.24</td>
<td>268</td>
</tr>
</tbody>
</table>
Comparing Cyclomatic Complexity: Android and Symbian

- Bug density in both the systems is significantly low
- Low average CC due to default functions
- High max CC due to inlining and macros
- Max CC in Android Kernel (4973) is much higher than in Symbian (1470)

In a Nutshell

- Most of the bugs are permanent in nature suggesting immature codebase
- Kernel in both systems is robust. More rigorous testing is needed for middleware.
- Failures in Dev tools, Web, Multimedia, and Doc-Install are common in both systems
- Customizability does lead to significant fraction of bugs
How Robust is Input Validation in Android? (with Fahad Arshad)

• Test various components in Android with random input
  – Activity
  – Services
  – Broadcast Receivers
• Send random messages to these components
  – Monitor stack trace from logcat

Activities: Search a Contact

• Main
• Search
• Display Contact
• Activities
  – UI component
**Intents**

- Intent: abstract operation to be performed
- Components Interact using **Intent messages**
- Intent-filter: component advertise Intents
- Intent Resolution
  - Caller calls callee by component name
  - Runtime determines callee based on Intent

<table>
<thead>
<tr>
<th>INTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌹 Component Name</td>
</tr>
<tr>
<td>🌹 Action</td>
</tr>
<tr>
<td>🌹 Data</td>
</tr>
<tr>
<td>🌹 Category</td>
</tr>
<tr>
<td>🌹 Extras</td>
</tr>
</tbody>
</table>

**Fuzzing Methodology**

- **IntentFuzzer**
  - Send random Intent messages to these components
  - Monitor stack trace
- **Crash ➔ Uncaught Exception**
Exception Handling Errors

<table>
<thead>
<tr>
<th>Component Type</th>
<th>No of Components Tested</th>
<th>No of Components Crashed</th>
<th>Type of Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast Receiver</td>
<td>42</td>
<td>8</td>
<td>NullPointerException</td>
</tr>
<tr>
<td>Services</td>
<td>27</td>
<td>3</td>
<td>NullPointerException</td>
</tr>
<tr>
<td>Activities Round 1</td>
<td>294</td>
<td>15</td>
<td>NullPointerException</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>ClassNotFoundException</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>IllegalArgumentException</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>ActivityNotFoundException</td>
</tr>
<tr>
<td>Activities Round 2</td>
<td>294</td>
<td>10</td>
<td>NullPointerException</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>ClassNotFoundException</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>IllegalArgumentException</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>ActivityNotFoundException</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>UnsupportedOperationException</td>
</tr>
</tbody>
</table>

Detected 36 Bugs

Security Concerns

- 4 of 36 detected bugs caused Android system process (android.server.ServerThread) to crash
- No additional permission was needed to run IntentFuzzer
  - Was able to run activities under privileged process
- App developers must be careful when dealing with Intents
  - Exception handling is a must!
**System Crash**

```java
I/ActivityManager( 62): Starting activity: Intent { act=ACTION_PACKAGE
android/.accounts.GrantCredentialsPermissionActivity }
W/dalvikvm( 62): threadid=7: thread exiting with uncaught exception (gr
E/AndroidRuntime( 62): *** FATAL EXCEPTION IN SYSTEM PROCESS: android.o
E/AndroidRuntime( 62): Caused by: java.lang.NullPointerException
E/AndroidRuntime( 62): at android.accounts.GrantCredentialsPermi
E/AndroidRuntime( 62): ... 6 more
I/Zygote ( 33): Exit zygote because system server (62) has terminated
```

```java
57 final Bundle extras = getIntent().getExtras();
58 mAccount = extras.getParcelable(EXTRAS_ACCOUNT);
```

**Conclusion**

- Input validation in Android needs more attention
- Intent passing and default security permissions are a concern
- Development tools, Web browser, Multimedia need to be more robust
- Both Android and Symbian show similar fault manifestation
Looking Forward

• Evaluation of Inter Component Communication in Android
  – Can the detected bugs be exploited?
• “Mobile phones are more personal than personal computers”
  – What are the privacy implications?
• Smartphones have lesser physical security
  – Encryption vs. usability

Thanks

Questions?