Taming Android Fragmentation: Characterizing and Detecting Compatibility Issues for Android Apps

Lili Wei
Yepang Liu
S.C. Cheung

The Hong Kong University of Science and Technology

05 September 2016
Android: Popular and Fast-Evolving System

- Fast-evolving: multiple releases every year
Android: Popular and Fast-Evolving System

- Fast-evolving: multiple releases every year
- Many different system versions in use:

https://developer.android.com/about/dashboards/index.html
open source project

- HTC
- ZTE
- Lenovo
- Acer
- Sony
- Huawei
- Samsung
- Motorola
- LG
- Asus
Huge Number of Device Models

Over 24,000 distinct device models

http://opensignal.com/reports/2015/08/android-fragmentation/
Fragmented Android Ecosystem
Fragmented Android Ecosystem
Fragmented Android Ecosystem
Ensuring App Compatibility Is Difficult

• App developers have to optimize UI of their apps to fit diversified screen sizes
Ensuring App Compatibility is Difficult

- App developers have to optimize UI of their apps to fit diversified screen sizes
- An app behaves differently across devices

http://gameovenstudios.com/bounden-on-android-delayed/
Ensuring App Compatibility is Difficult

- App developers have to optimize UI of their apps to fit diversified screen sizes
- An app behaves differently across devices

Fragmentation-induced compatibility issues

http://gameovenstudios.com/bounden-on-android-delayed/
Testing Compatibility Issues

• To fully test compatibility issues is difficult with huge search space
  • Combination of three dimensions
Testing Compatibility Issues

- To fully test compatibility issues is difficult with huge search space
  - Combination of three dimensions

![Diagram showing API Level and Device Model dimensions]
Testing Compatibility Issues

• To fully test compatibility issues is difficult with huge search space
  • Combination of **three dimensions**
Testing Compatibility Issues

• To fully test compatibility issues is difficult with huge search space
  • Combination of three dimensions
Existing Work

- Understanding Android fragmentation
  - D. Han et al. [WCRE’ 2012]
  - T. McDonnell et al. [ICSM’ 2013]

- Prioritize devices to test on:
  - H. Khalid et al. [FSE’ 2014]
  - X. Lu et al. [ICSE’ 2016]
Existing Work

• Understanding Android fragmentation
  • D. Han et al. [WCRE’ 2012]
  • T. McDonnell et al. [ICSM’ 2013]

• Prioritize devices to test on:
  • H. Khalid et al. [FSE’ 2014]
  • X. Lu et al. [ICSE’ 2016]

None of the existing studies dug into the code level:
  • To understand compatibility issues
  • To detect compatibility issues
Our Goal

Understand compatibility issues at code level
Our Goal

Understand compatibility issues at code level

Detect compatibility issues
Research Questions

• RQ1: Issue type and root cause
  • What are the common types of compatibility issues in Android apps? What are their root causes?
Research Questions

• **RQ1: Issue type and root cause**
  • What are the common types of compatibility issues in Android apps? What are their root causes?

• **RQ2: Issue fixing**
  • How do Android developers fix compatibility issues in practice? Are there any common patterns?
Dataset Collection

27 popular, large-scale, well-maintained candidate apps, which have public issue tracking systems
27 popular, large-scale, well-maintained candidate apps, which have public issue tracking systems

Apps satisfying these criterion are:
• More likely to be affected by different kinds of compatibility issues
• With traceable issue fix and discussions to study
## Keyword Search

<table>
<thead>
<tr>
<th>device</th>
<th>compatible</th>
<th>compatibility</th>
<th>samsung</th>
</tr>
</thead>
<tbody>
<tr>
<td>lge</td>
<td>sony</td>
<td>moto</td>
<td>lenovo</td>
</tr>
<tr>
<td>asus</td>
<td>zte</td>
<td>google</td>
<td>htc</td>
</tr>
<tr>
<td>huawei</td>
<td>xiaomi</td>
<td></td>
<td>android.os.build</td>
</tr>
</tbody>
</table>
VLC r-8e31d57:
Blind fix for mountpoint issues with some Samsung devices

if(line.contains("vfat") || line.contains("exfat") ||
  line.contains("/mnt") || line.contains("/Removable")) {
  + line.contains("sdcardfs") || line.contains("/mnt") ||
  + line.contains("/Removable")) {
## VLC r-8e31d57:
Blind fix for mountpoint issues with some **Samsung** devices

```
if(line.contains("vfat") || line.contains("exfat") ||
-    line.contains("/mnt") || line.contains("/Removable")}) {
+    line.contains("sdcardfs") || line.contains("/mnt") ||
+    line.contains("/Removable")}) {
```

## CSipSimple r-af0ceeda:
Fixes issue 498.

```
+    //HTC evo 4G
+    if(android.os.Build.PRODUCT.equalsIgnoreCase("htc_supersonic"))
+        return true;
```
 Dataset Collection

27 popular, large-scale, well-maintained candidate apps, which have public issue tracking systems

All 27 apps contain potential revisions related to compatibility issues
Dataset Collection

27 **popular, large-scale, well-maintained** candidate apps, which have **public issue tracking systems**

Five apps with top number of potential issue-related code revisions
Dataset Collection

Google Code  F-Droid  GitHub

27 popular, large-scale, well-maintained candidate apps, which have public issue tracking systems

Five apps with top number of potential issue-related code revisions

Over 1800 code revisions found in the five apps need to be manually inspected
Dataset Collection

27 popular, large-scale, well-maintained candidate apps, which have public issue tracking systems

Five apps with top number of potential issue-related code revisions (>1800 revisions)

191 compatibility issues
## Selected Apps

<table>
<thead>
<tr>
<th>App Name</th>
<th>Category</th>
<th>Rating</th>
<th>Downloads</th>
<th>KLOC</th>
<th>#Revisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSipSimple</td>
<td>Communication</td>
<td>4.3/5.0</td>
<td>1M - 5M</td>
<td>59.2</td>
<td>1,778</td>
</tr>
<tr>
<td>AnkiDroid</td>
<td>Education</td>
<td>4.5/5.0</td>
<td>1M - 5M</td>
<td>58.1</td>
<td>8,282</td>
</tr>
<tr>
<td>K-9 Mail</td>
<td>Communication</td>
<td>4.3/5.0</td>
<td>5M - 10M</td>
<td>86.7</td>
<td>6,116</td>
</tr>
<tr>
<td>VLC</td>
<td>Media &amp; Video</td>
<td>4.3/5.0</td>
<td>10M - 50M</td>
<td>28.3</td>
<td>6,868</td>
</tr>
<tr>
<td>AnySoftKeyboard</td>
<td>Tools</td>
<td>4.3/5.0</td>
<td>1M - 5M</td>
<td>70.7</td>
<td>2,788</td>
</tr>
</tbody>
</table>
Data Analysis

Identified the issue-inducing APIs
Data Analysis

- Identified the issue-inducing APIs
- Recovered links between revisions and reports
Data Analysis

- Identified the issue-inducing APIs
- Recovered links between revisions and reports
- Collected related discussions from online forums
RQ1: Issue Type and Root Cause
RQ1: Issue Type and Root Cause

- Device-specific issues
  - Compatibility issues can only be triggered on certain device models

- Non-device-specific issue
  - Compatibility issues can be triggered under certain API level independent from device model
RQ1: Issue Type and Root Cause

Non-device-specific issues
Can be triggered under certain API level

Device-specific issues
Can only be triggered on certain device models
Device-Specific Issue

- Device-specific Issue: 59%
- Driver Implementation: 29%
- OS Customization: 19%
- Hardware Composition: 11%
Device-Specific Issues

• Problematic driver implementation
  • Different driver implementation can cause inconsistent app behavior across devices.
Device-Specific Issues

• Problematic driver implementation
  • Different driver implementation can cause inconsistent app behavior across devices.

• OS customization
  • Manufacturer-customized systems may not fully comply with the official specifications and thus cause compatibility issues
Device-Specific Issues

- Problematic driver implementation
  - Different driver implementation can cause inconsistent app behavior across devices.

- OS customization
  - Manufacturer-customized systems may not fully comply with the official specifications and thus cause compatibility issues.

- Hardware composition
  - Diversified hardware composition of different device models can cause compatibility issues.
Device-Specific Issues

• Problematic driver implementation
  • Different driver implementation can cause inconsistent app behavior across devices.

Proximity sensor: Report the distance between device and its surrounding object
Device-Specific Issues

CSipSimple: An Android client app for internet calls

Proximity sensor in CSipSimple: Used to detect the distance between user’s face and the device.
Device-Specific Issues

CSipSimple:
An Android client app for internet calls

Proximity sensor in CSipSimple:
*Used to detect the distance between user’s face and the device.*
Device-Specific Issues

CSipSimple:
An Android client app for internet calls

Proximity sensor in CSipSimple:
*Used to detect the distance between user’s face and the device.*

CSipSimple issue 353:
Samsung SPH-M900’s proximity sensor API reports a value inversely proportional to the real distance
Device-Specific Issues

CSipSimple: An Android client app for internet calls

Proximity sensor in CSipSimple: Used to detect the distance between user’s face and the device.

CSipSimple issue 353: Samsung SPH-M900’s proximity sensor API reports a value inversely proportional to the real distance
Device-Specific Issues

CSipSimple: An Android client app for internet calls

Proximity sensor in CSipSimple: 
*Used to detect the distance between user’s face and the device.*

CSipSimple issue 353:
Samsung SPH-M900’s proximity sensor API reports a value inversely proportional to the real distance
Device-Specific Issues

**CSipSimple:**
An Android client app for internet calls

Proximity sensor in CSipSimple:
*Used to detect the distance between user’s face and the device.*

**CSipSimple issue 353:**
Samsung SPH-M900’s proximity sensor API reports a value inversely proportional to the real distance
Device-Specific Issues

• Peculiar hardware composition

SD card variations
Device-Specific Issues

- Peculiar hardware composition

SD card variations
Device-Specific Issues

• Peculiar hardware composition

SD card variations
Device-Specific Issues

• Peculiar hardware composition

SD card variations
Device-Specific Issues

• Peculiar hardware composition

SD card variations

Mount points vary across devices
Device-Specific Issues

VLC:
A popular Android video player

SD card in VLC:
VLC can play videos stored in the SD card
Device-Specific Issues

VLC: A popular Android video player

SD card in VLC: VLC can play videos stored in the SD card

VLC specifically handle the SD card variations to ensure support of different device models
Non-Device-Specific Issue

- Non-Device-Specific Issue: 41%
- API Evolution: 35%
- Original System Bug: 6%
Non-Device-Specific Issues

• Android platform API evolution
  • Evolving Android system with API introduction, deprecation and behavior modification causes compatibility issues
Non-Device-Specific Issues

• Android platform API evolution
  • Evolving Android system with API introduction, deprecation and behavior modification causes compatibility issues

• Original Android system bugs
  • Bugs in the system got fixed in newer version while still existing in older versions can cause compatibility issues.
Non-Device-Specific Issues

AnkiDroid pull request 130

API (Added in API Level 16):
`SQLiteDatabase.disableWriteAheadLogging()`
Non-Device-Specific Issues

AnkiDroid pull request 130

API (Added in API Level 16):
SQLiteDatabase.disableWriteAheadLogging()
Non-Device-Specific Issues

AnkiDroid pull request 130

API (Added in API Level 16):
SQLiteDatabase.disableWriteAheadLogging()
RQ2: Issue Fixing

• Patch Complexity
• Common Patching Patterns
RQ2: Issue Fixing

- Patch Complexity
  - Simple
  - Small

```java
1. + if (android.os.Build.VERSION.SDK_INT >= 16) {
2.     sqLiteDatabase.disableWriteAheadLogging();
3. + }
```
RQ2: Issue Fixing

• Patch Complexity
  • To debug and find a valid fix is not easy

CSipSimple issue 2436:
“HTC has closed source this contact/call log app which makes things almost impossible to debug for me.”
RQ2: Issue Fixing

• Common patching patterns
  • Check device information (71.7%)
RQ2: Issue Fixing

- Common patching patterns
  - Check device information (71.7%)
  - Check availability of software and hardware components (7.3%)
RQ2: Issue Fixing

• Common patching patterns
  • Check device information (71.7%)
  • Check availability of software and hardware components (7.3%)  
  • App-specific workarounds
RQ2: Issue Fixing

• Common patching patterns
  • Check device information (71.7%)
  • Check availability of software and hardware components (7.3%)
  • App-specific workarounds

Locating the root cause of compatibility issues is difficult. Whereas, issue fixes are usually simple and demonstrate common patterns.
Applications of the Empirical Findings

• Prioritize testing efforts
Applications of the Empirical Findings

• Prioritize testing efforts
Applications of the Empirical Findings

• Prioritize testing efforts
Applications of the Empirical Findings

• Detect compatibility issues
API-Context Pair Model

• Common compatibility issue pattern

Compatibility issues are triggered by the improper use of an Android API (issue-inducing API) in a problematic software or hardware environment (issue-triggering context)
API-Context Pair Model

• Common compatibility issue pattern

**API-Context Pair Model:**
Each pattern of compatibility issue is modeled as a pair of *issue-inducing API* and *issue-triggering context*
API-Context Pair Model

• Common compatibility issue pattern

API-Context Pair Model:
Each pattern of compatibility issue is modeled as a pair of issue-inducing API and issue-triggering context

Context: conjunction of device model, device brand or API level, etc.
API-Context Pair Model

- Common compatibility issue pattern

**API-Context Pair Model:**
Each pattern of compatibility issue is modeled as a pair of **issue-inducing API** and **issue-triggering context**

**API:** `SQLiteDatabase.disableWriteAheadLogging()`
**Context:** `API_level < 16 ∧ Dev_model != "Nook HD"`
FicFinder: Detecting Compatibility Issues

FicFinder:
Fragmen
tation-
Induced C ompatibility Issue Finder
FicFinder: Detecting Compatibility Issues

- Android app (.apk or .class)
- List of API-context pairs

Analysis Report
FicFinder: Detecting Compatibility Issues

Android app (.apk or .class)

List of API-context pairs

Backward slicing analysis
Implemented on top of Soot
25 selected API-context pairs

Analysis Report
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }

    db.disableWriteAheadLogging();
}
```

**API:** SQLiteDatabase.disableWriteAheadLogging()

**Context:** API_level < 16 ∧ Dev_model != "Nook HD"
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    db.disableWriteAheadLogging();
}
```

**API:** SQLiteDatabase.disableWriteAheadLogging()

**Context:** API_level < 16 ∧ Dev_model != "Nook HD"
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    db.disableWriteAheadLogging();
}
```

AndroidManifest.xml
```xml
<uses-sdk
    android:minSdkVersion="15"
/>```

**API:** SQLiteDatabase.disableWriteAheadLogging()

**Context:** API_level < 16 ∧ Dev_model != "Nook HD"
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    db.disableWriteAheadLogging();
}
```

**API:** SQLiteDatabase.disableWriteAheadLogging()

**Context:** API_level < 16 ∧ Dev_model != "Nook HD"

**AndroidManifest.xml**
```
<uses-sdk
    android:minSdkVersion="15" />
```

**Slice:**
```
db.disableWriteAheadLogging()
```
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    db.disableWriteAheadLogging();
}
```

**API:** `SQLiteDatabase.disableWriteAheadLogging()`  
**Context:** `API_level < 16 ∧ Dev_model != "Nook HD"`
Example

void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    db.disableWriteAheadLogging();
}

API: SQLiteDatabase.disableWriteAheadLogging()
Context: API_level < 16 ∧ Dev_model != "Nook HD"
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }

    db.disableWriteAheadLogging();
}
```

**API:** SQLiteDatabase.disableWriteAheadLogging()

**Context:** API_level < 16 ∧ Dev_model != "Nook HD"
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    db.disableWriteAheadLogging();
}
```

API: SQLiteDatabase.disableWriteAheadLogging()
Context: API_level < 16 ∧ Dev_model != "Nook HD"

AndroidManifest.xml
```xml
<uses-sdk
    android:minSdkVersion="15"
/>```

Slice:
```java
db.disableWriteAheadLogging()
SQLiteDatabase db = getDatabase()
```
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    db.disableWriteAheadLogging();
}
```

**API:**  SQLiteDatabase.disableWriteAheadLogging()

**Context:**  API_level < 16 ∧ Dev_model != "Nook HD"

---

**AndroidManifest.xml**

```
<uses-sdk
    android:minSdkVersion="15"
/>
```

**Slice:**

```
SQLiteDatabase db = getDatabase()
```
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    if (isJellyBeanOrLater() || isNookHD) {
        db.disableWriteAheadLogging();
    }
}

boolean isJellyBeanOrLater() {
    return android.os.Build.SDK_VERSION >= 16;
}

boolean isNookHD {
    return android.os.Build.BRAND.equals("NOOK")
    && android.os.Build.PRODUCT.equals("HDplus")
    && android.os.Build.DEVICE.equals("ovation");
}
```

AndroidManifest.xml
```
<uses-sdk
    android:minSdkVersion="15"
/>
```

Slice:
```
db.disableWriteAheadLogging()
SQLiteDatabase db = getDatabase()
```

**API:** SQLiteDatabase.disableWriteAheadLogging()

**Context:** API_level < 16 ∧ Dev_model != "Nook HD"
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    if (isJellyBeanOrLater() || isNookHD) {
        db.disableWriteAheadLogging();
    }
}
```

```xml
AndroidManifest.xml
<uses-sdk
    android:minSdkVersion="15"
/>
```

```java
boolean isJellyBeanOrLater() {
    return android.os.Build.SDK_VERSION >= 16;
}
```

```java
boolean isNookHD {
    return android.os.Build.BRAND.equals("NOOK")
    && android.os.Build.PRODUCT.equals("HDplus")
    && android.os.Build.DEVICE.equals("ovation");
}
```

**API:** SQLiteDatabase.disableWriteAheadLogging()

**Context:** API_level < 16 \&\& Dev_model != "Nook HD"
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    if (isJellyBeanOrLater() || isNookHD) {
        db.disableWriteAheadLogging();
    }
}

boolean isJellyBeanOrLater() {
    return android.os.Build.SDK_VERSION >= 16;
}

boolean isNookHD {
    return android.os.Build.BRAND.equals("NOOK")
        && android.os.Build.PRODUCT.equals("HDplus")
        && android.os.Build.DEVICE.equals("ovation");
}

AndroidManifest.xml
<uses-sdk
    android:minSdkVersion="15"
/>

Slice:
SQLiteDatabase db = getDatabase()
if(isJellyBeanOrLater()||isNookHD)
return android.os.Build.SDK_VERSION >= 16
return android.os.Build.BRAND.equals("NOOK")
    && android.os.Build.PRODUCT.equals("HDplus")
    && android.os.Build.DEVICE.equals("ovation")

API: SQLiteDatabase.disableWriteAheadLogging()
Context: API_level < 16 ∧ Dev_model != "Nook HD"
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    if (isJellyBeanOrLater() || isNookHD) {
        db.disableWriteAheadLogging();
    }
}

boolean isJellyBeanOrLater() {
    return android.os.Build.SDK_VERSION >= 16;
}

boolean isNookHD {
    return android.os.Build.BRAND.equals("NOOK")
        && android.os.Build.PRODUCT.equals("HDplus")
        && android.os.Build.DEVICE.equals("ovation");
}
```

AndroidManifest.xml

```xml
<uses-sdk
    android:minSdkVersion="15"
/>
```

Slice:

```java
db.disableWriteAheadLogging()

SQLiteDatabase db = getDatabase()
if(isJellyBeanOrLater()||isNookHD)
return android.os.Build.SDK_VERSION >= 16

return android.os.Build.BRAND.equals("NOOK")
    && android.os.Build.PRODUCT.equals("HDplus")
    && android.os.Build.DEVICE.equals("ovation");
```

API: SQLiteDatabase.disableWriteAheadLogging()

**Context:** API_level < 16 ∧ Dev_model != "Nook HD"
Example

```java
void disableWAL() {
    SQLiteDatabase db = getDatabase();
    if (db.inTransaction()) {
        db.endTransaction();
    }
    if (isJellyBeanOrLater() || isNookHD) {
        db.disableWriteAheadLogging();
    }
}

boolean isJellyBeanOrLater() {
    return android.os.Build.SDK_VERSION >= 16;
}

boolean isNookHD {
    return android.os.Build.BRAND.equals("NOOK")
    && android.os.Build.PRODUCT.equals("HDplus")
    && android.os.Build.DEVICE.equals("ovation");
}
```

**API:**  SQLiteDatabase-disableWriteAheadLogging()

**Context:**  API_level < 16 ∧ Dev_model != "Nook HD"

AndroidManifest.xml

```xml
<uses-sdk
    android:minSdkVersion="15"
/>
```

Slice:

```java
db.disableWriteAheadLogging()
SQLiteDatabase db = getDatabase()
if(isJellyBeanOrLater()||isNookHD)
  return android.os.Build.SDK_VERSION >= 16
  return android.os.Build.BRAND.equals("NOOK")
  && android.os.Build.PRODUCT.equals("HDplus")
  && android.os.Build.DEVICE.equals("ovation")
```
## Evaluation

- **Subject apps**: latest version of 27 popular, actively-maintained Android apps
  - Covering 10 different categories
  - At least 5K downloads

<table>
<thead>
<tr>
<th>App name</th>
<th>Category</th>
<th>KLOC</th>
<th>Rating</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectBot</td>
<td>Communication</td>
<td>23.0</td>
<td>4.6</td>
<td>1M – 5M</td>
</tr>
<tr>
<td>AntennaPod</td>
<td>Media &amp; Video</td>
<td>65.0</td>
<td>4.5</td>
<td>100K – 500K</td>
</tr>
<tr>
<td>c:geo</td>
<td>Entertainment</td>
<td>78.8</td>
<td>4.4</td>
<td>1M – 5M</td>
</tr>
<tr>
<td>AnySoftKeyboard</td>
<td>Tools</td>
<td>70.7</td>
<td>4.4</td>
<td>1M – 5M</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Effectiveness of FicFinder

• Configure FicFinder to report both warnings and good practices
Effectiveness of FicFinder

• Configure FicFinder to report both warnings and good practices
• FicFinder reported 51 warnings and 79 good practices
Effectiveness of FicFinder

- Configure FicFinder to report both warnings and good practices
- FicFinder reported 51 warnings and 79 good practices

46 of the 51 warnings are true positives  (Precision: 90.2%)
Effectiveness of FicFinder

• Submitted 14 bug reports
  • Bug reports included issue-related discussions, API guides and possible fixes in the bug reports to help developers diagnose the issues
Effectiveness of FicFinder

• Submitted 14 bug reports
  • Bug reports included issue-related discussions, API guides and possible fixes in the bug reports to help developers diagnose the issues

  8 of the 14 bug reports were acknowledged by app developers.
  5 issues were quickly fixed.
Usefulness of FicFinder

• Knowledge of API-context pairs extracted from our empirical study can be transferred across different apps

AnySoftKeyboard Issue 639:
Rule originally extracted from K9 Mail

K9 Mail Issue 1237:
Rule originally extracted from CSipSimple
Usefulness of FicFinder

- **4** of the 5 fixed issues were fixed by adopting our suggested fixes.

iSsO (1Sheeld developer)

Thanks, we will do it as you proposed in our next release. :) Please let us know if you found more of these issues.
Usefulness of FicFinder

- 4 of the 5 fixed issues were fixed by adopting our suggested fixes.
- The other issue was fixed with an alternative patch, which was semantically equivalent to our suggested one.

Thanks, we will do it as you proposed in our next release. :)
Please let us know if you found more of these issues.

iSsO (1Sheeld developer)
Conclusion

• First large-scale empirical study of Android compatibility issues at code level
• API-context pair to model compatibility issues
• A static analysis tool, FicFinder, to detect fragmentation-induced compatibility issues
Future Work

• Automate API-context pair extraction
• Continue to improve the effectiveness of FicFinder
Thank you!

More information can be found in our paper
Or visit FicFinder’s homepage:
http://sccpu2.cse.ust.hk/ficfinder/