Automating GUI Testing for Android Applications

Cuixiong(Tony) Hu                    Iulian Neamtiu

Amazon Inc.                        University of California, Riverside
Android Gaining Market Share
Android Gaining Market Share

- Android: 36%
- Symbian: 27.4%
- iOS: 16.8%
- RIM: 12.9%
- Windows: 3.6%
- Others: 3.3%

Timeline:
- Q1'09
- Q3'09
- Q1'10
- Q3'10
- Q1'11
More and More New Android Applications Released
More and More New Android Applications Released

Tuesday, May 24, 2011
Motivation
Motivation

✦ Android platform gaining traction
Motivation

✦ Android platform gaining traction
  Correctness of Android applications is CRITICAL (mobile banking, business mgmt.)
Motivation

- Android platform gaining traction
  Correctness of Android applications is CRITICAL
  (mobile banking, business mgmt.)

What’s novel about the Android platform?
Motivation

✦ Android platform gaining traction
   Correctness of Android applications is CRITICAL
   (mobile banking, business mgmt.)

✦ Novelty of Android development model

What’s novel about the Android platform?
Motivation

- Android platform gaining traction
  Correctness of Android applications is CRITICAL
  (mobile banking, business mgmt.)

- Novelty of Android development model

  Android-specific bug study

What’s novel about the Android platform?
Motivation

- Android platform gaining traction
  Correctness of Android applications is CRITICAL (mobile banking, business mgmt.)

- Novelty of Android development model

- Android-specific bug study

- Android-specific verification tools

What’s novel about the Android platform?
Motivation

✦ Android platform gaining traction
  Correctness of Android applications is CRITICAL
  (mobile banking, business mgmt.)

✦ Novelty of Android development model

Android-specific bug study

✦ Android-specific verification tools

Automated approach for test generation and runtime verification for Android applications

What’s novel about the Android platform?
What is Android

✦ Open source platform for mobile devices, started October 2008
✦ A complete software stack
  ✦ OS
  ✦ Middleware
  ✦ Applications
✦ Based on Java
Components of Android Applications

✦ Activity
  ✦ Control application windows

✦ Services
  ✦ Run in the background for an indefinite period of time

✦ Broadcast Receivers
  ✦ React to a broadcast information, i.e., low battery

✦ Content Providers
  ✦ Store content, allow sharing with other apps
Architecture of Android
Architecture of Android

Application
Architecture of Android

- Music Player
- Web Browser
- Mobile Banking

Application
Architecture of Android

Application Framework

- Music Player
- Web Browser
- Mobile Banking

Application
Architecture of Android

Application
- Music Player
- Web Browser
- Mobile Banking

Application Framework
- Activity Manager
- Content Provider
Architecture of Android

Application

- Music Player
- Web Browser
- Mobile Banking

Application Framework

- Activity Manager
- Content Provider

Data
Architecture of Android

Application Framework

Activity Manager

Content Provider

Music Player

Web Browser

Mobile Banking

Application

Android Runtime

Data
Architecture of Android

Application

Music Player
Web Browser
Mobile Banking

Activity Manager
Content Provider

Application Framework

Android Runtime

Data

Tuesday, May 24, 2011
Architecture of Android

Application

Music Player
Web Browser
Mobile Banking

Application Framework

Activity Manager
Content Provider

Android Runtime

Dalvik VM
Architecture of Android

Application
- Music Player
- Web Browser
- Mobile Banking
- Activity Manager
- Content Provider

Application Framework

Android Runtime
- Dalvik VM

Linux Kernel

Tuesday, May 24, 2011
Android development model is unique
Roadmap

Android development model is unique

Android-specific, novel kinds of bugs
Android development model is unique

Android-specific, novel kinds of bugs

Bug study and bug categorization
## Applications Examined

<table>
<thead>
<tr>
<th>Program</th>
<th>First Release</th>
<th># of updates</th>
<th># of bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andoku</td>
<td>06/2009</td>
<td>382</td>
<td>1</td>
</tr>
<tr>
<td>GuessTheNumber</td>
<td>02/2009</td>
<td>71</td>
<td>2</td>
</tr>
<tr>
<td>Delicious</td>
<td>02/2009</td>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td>MonolithAndroid</td>
<td>11/2008</td>
<td>167</td>
<td>7</td>
</tr>
<tr>
<td>Opensudoku</td>
<td>04/2009</td>
<td>393</td>
<td>7</td>
</tr>
<tr>
<td>CMIS</td>
<td>01/2010</td>
<td>41</td>
<td>8</td>
</tr>
<tr>
<td>DealDroid</td>
<td>03/2009</td>
<td>164</td>
<td>10</td>
</tr>
<tr>
<td>Skylight1</td>
<td>09/2009</td>
<td>709</td>
<td>10</td>
</tr>
<tr>
<td>Rokon</td>
<td>09/2009</td>
<td>362</td>
<td>29</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>08/2008</td>
<td>508</td>
<td>79</td>
</tr>
</tbody>
</table>
## Bug Study Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Activity</th>
<th>Event</th>
<th>Type</th>
<th>Unhandled Exception</th>
<th>API</th>
<th>I/O</th>
<th>Concurrency</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylight1</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>CMIS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Delicious</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>57</td>
<td>79</td>
</tr>
<tr>
<td>DealDroid</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Rokon</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>14</td>
<td>29</td>
</tr>
<tr>
<td>Andoku</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Opensudoku</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>GuessTheNumber</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>MonolithAndroid</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8</strong></td>
<td><strong>21</strong></td>
<td><strong>4</strong></td>
<td><strong>11</strong></td>
<td><strong>4</strong></td>
<td><strong>7</strong></td>
<td><strong>1</strong></td>
<td><strong>102</strong></td>
<td><strong>158</strong></td>
</tr>
</tbody>
</table>

Tuesday, May 24, 2011
## Bug Study Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Activity</th>
<th>Event</th>
<th>Type</th>
<th>Unhandled Exception</th>
<th>API</th>
<th>I/O</th>
<th>Concurrency</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylight1</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>CMIS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Delicious</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>57</td>
<td>79</td>
</tr>
<tr>
<td>DealDroid</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Rokon</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>14</td>
<td>29</td>
</tr>
<tr>
<td>Andoku</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Opensudoku</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>GuessTheNumber</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>MonolithAndroid</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8</strong></td>
<td><strong>21</strong></td>
<td><strong>4</strong></td>
<td><strong>11</strong></td>
<td><strong>4</strong></td>
<td><strong>7</strong></td>
<td><strong>1</strong></td>
<td><strong>102</strong></td>
<td><strong>158</strong></td>
</tr>
<tr>
<td>Program</td>
<td>Activity</td>
<td>Event</td>
<td>Type</td>
<td>Unhandled Exception</td>
<td>API</td>
<td>I/O</td>
<td>Concurrency</td>
<td>Other</td>
<td>Total</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>-------</td>
<td>------</td>
<td>---------------------</td>
<td>-----</td>
<td>-----</td>
<td>-------------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Skylight1</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>CMIS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Delicious</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>57</td>
</tr>
<tr>
<td>DealDroid</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Rokon</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>Andoku</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Opensudoku</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>GuessTheNumber</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>MonolithAndroid</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8</strong></td>
<td><strong>21</strong></td>
<td><strong>4</strong></td>
<td><strong>11</strong></td>
<td><strong>4</strong></td>
<td><strong>7</strong></td>
<td><strong>1</strong></td>
<td><strong>0</strong></td>
<td><strong>102</strong></td>
</tr>
</tbody>
</table>

**How to detect?**

Tuesday, May 24, 2011
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring

Application
- Activity1
- Activity2
- Activity3
- Activity4

JUnit
- Initial Condition
- GUI
- State Management
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring
Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring

Application
- Activity1
- Activity2
- Activity3
- Activity4

JUnit
- Events
- Initial Condition
- GUI
- Events
- State Management
- Events

Monkey
- Random
- Deterministic
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring
Testing Techniques

Combination of multiple methods:
1. Automatic event generation
2. Test case generation
3. Post-run application performance monitoring
Automatic Log File Analysis

- Activity and event bugs: state machine-based analysis
- Runtime type errors: look for exceptions
State machine used to detect activity and event bugs
State machine used to detect activity and event bugs
State Machine-based Analysis

State machine used to detect activity and event bugs
Correct Pattern:

**State machine used to detect activity and event bugs**
State Machine-based Analysis

Correct Pattern:

State machine used to detect activity and event bugs
State Machine-based Analysis

Correct Pattern:

State machine used to detect activity and event bugs
Correct Pattern:

State machine used to detect activity and event bugs
Correct Pattern:

State machine used to detect activity and event bugs
State Machine-based Analysis

Correct Pattern:

State machine used to detect activity and event bugs
State Machine-based Analysis

State machine used to detect activity and event bugs
State Machine-based Analysis

Incorrect Pattern:

State machine used to detect activity and event bugs
Incorrect Pattern:

State machine used to detect activity and event bugs
Incorrect Pattern:

State machine used to detect activity and event bugs
Incorrect Pattern:

State machine used to detect activity and event bugs
State Machine-based Analysis

Incorrect Pattern:

State machine used to detect activity and event bugs
Crash Example

1. I/Starting activity Intent {action=android.intent.category.HOME}
2. D/:Shutting down VM
Crash Example

I/Starting activity Intent {action=android.intent.category.HOME}
D/:Shutting down VM

Sorry!
The application ConnectBot (process org.connectbot) has stopped unexpectedly. Please try again.
Exception Analysis

ClassCastException in the log file

1. I/Starting activity: Intent {comp={org.connectbot/org.connectbot.SettingsActivity}}
2. E/java.lang.RuntimeException: java.lang.ClassCastException
Using Our Tool as Debugging Aid

Unrolled stack helps developers reproduce bugs and pinpoint errors

```
E/AndroidRuntime( 190):
    at org.connectbot.SettingsActivity.onCreate( SettingsActivity.java:29)
E/AndroidRuntime( 190):
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2364)
I/ActivityManager(  52): Process org.connectbot (pid 190) has died.
D/AndroidRuntime( 211): Shutting down VM
W/dalvikvm( 211): threadid=3: thread exiting with uncaught exception (group=0x4001aa28)
E/AndroidRuntime( 211): Uncaught handler: thread main exiting due to uncaught exception
```
Using Our Tool as Debugging Aid

Unrolled stack helps developers reproduce bugs and pinpoint errors

```
1  E/AndroidRuntime( 190): onCreate() (SettingsActivity.java:29)
2  E/AndroidRuntime( 190): onCreate()
3  E/AndroidRuntime( 190): onCreate()
4  E/AndroidRuntime( 190): onCreate()
5  E/AndroidRuntime( 190): onCreate()
6  E/AndroidRuntime( 190): onCreate()
7  E/AndroidRuntime( 190): onCreate()
8  E/AndroidRuntime( 190): onCreate()
```

```
I/ActivityManager(  52): Process org.connectbot (pid 190) has died
D/AndroidRuntime( 211): Shutting down VM
W/dalvikvm( 211): threadid=3: thread exiting with uncaught exception (group=0x4001aa28)
E/AndroidRuntime( 211): Uncaused handler: thread main exiting due to uncaught exception
```
## Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Activity</th>
<th>Event</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylight1</td>
<td>3 (3)</td>
<td>2 (5)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>CMIS</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Delicious</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>2 (4)</td>
<td>8 (8)</td>
<td>2 (2)</td>
</tr>
<tr>
<td>DealDroid</td>
<td>1 (1)</td>
<td>1 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Rokon</td>
<td>0 (0)</td>
<td>6 (6)</td>
<td>2 (2)</td>
</tr>
<tr>
<td>Andoku</td>
<td>0 (1)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Opensudoku</td>
<td>1 (1)</td>
<td>1 (2)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>GuessTheNumber</td>
<td>1 (1)</td>
<td>1 (1)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>MonolithAndroid</td>
<td>0 (0)</td>
<td>2 (2)</td>
<td>0 (0)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8 (11)</strong></td>
<td><strong>21 (24)</strong></td>
<td><strong>4 (4)</strong></td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Activity</th>
<th>Event</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylight1</td>
<td>3 (3)</td>
<td>2 (5)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>CMIS</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Delicious</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>2 (4)</td>
<td>8 (8)</td>
<td>2 (2)</td>
</tr>
<tr>
<td>DealDroid</td>
<td>1 (1)</td>
<td>1 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Rokon</td>
<td>0 (0)</td>
<td>6 (6)</td>
<td>2 (2)</td>
</tr>
<tr>
<td>Andoku</td>
<td>0 (1)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Opensudoku</td>
<td>1 (1)</td>
<td>1 (2)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>GuessTheNumber</td>
<td>1 (1)</td>
<td>1 (1)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>MonolithAndroid</td>
<td>0 (0)</td>
<td>2 (2)</td>
<td>0 (0)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>8 (11)</strong></td>
<td><strong>21 (24)</strong></td>
<td><strong>4 (4)</strong></td>
</tr>
</tbody>
</table>
Conclusions

✦ Android applications prone to specific errors (activity, event)

✦ We perform a bug study and categorization on 10 Android applications

✦ New Android-specific verification techniques needed

✦ We implement a tool to detect activity/event/type bugs (Available on http://www.cs.ucr.edu/~huc)

✦ Our work proved effective for detecting Android bugs

✦ Our approach can be extended to capture other types of bugs like API, I/O, or concurrency errors