Smart Android GUI Testing Approaches

Yavuz Koroglu    Alper Sen

Department of Computer Engineering
Bogazici University, Istanbul/Turkey
yavuz.koroglu@boun.edu.tr
depend.cmpe.boun.edu.tr

November 6, 2017
Overview

1. Motivation
   - Android Usage Today
   - Fully-Automated Testing

2. Android GUI Testing
   - Android Background
   - Activity Life-Cycle
   - States and Actions
   - Remote Control DEMO

3. Monkey
   - Description
   - Pros & Cons

4. Monkey DEMO

5. Other Tools
   - Crashes
   - Coverage

6. Measures of Performance
   - AndroFrame
   - Reinforcement Learning
   - Test Case Mutation

7. Our Studies
   - Some Results
   - Future Work

8. Conclusions and Remarks
Android Usage Today

- We use phones 3 hours/day.
Android Usage Today

- We use phones 3 hours/day.
- We constantly get error messages.
Android Usage Today

- We use phones 3 hours/day.
- We constantly get error messages.
We use phones 3 hours/day.

We constantly get error messages.
We use phones 3 hours/day.

We constantly get error messages.
Fully-Automated Android GUI Testing

GUI Testing

- Click buttons,
- Fill textboxes,
- Drag & drop,
- Swipe,
- Toggle WiFi etc.

Automation is a MUST

> 2.2M Applications in the Android market.
Internal Structure of an Android Application (.APK File)

Structure Overview

1. Executable .DEX file,
2. AndroidManifest.xml, and
3. Other resources - pictures, sounds etc.

AndroidManifest.xml

- Activity Names,
- Launchable Activities, and
- Permissions.

.DEX File

- Formed by **Java classes**.
- Each class has methods.
Class Categories

1. **Activity:** Represents **different screens** of the application.
   - **Launchable Activity:** The **first** activity of the application.

2. **Service:** Represents tasks that runs in **background**. Started and stopped from activities.

3. **Content Provider:** Dynamically presents the information provided by various services to the activity.

4. **Broadcast Receiver:** Triggered by **external events** (SMS, GPS, clock timeout etc.) and activates specific code segments. Activities do **NOT** trigger them.

5. **Other Classes:** All other classes that inherit `java.lang.Object`.
Activity Life-Cycle

Properties of Activities

- Defaults expected to be overwritten.
- Developers depend on defaults.
- Error-prone.
Services, Content Providers, and Broadcast Receivers

Android Application Anatomy

- OS fires events.
- Broadcast receivers and the target activity receive the event.
- Event receivers trigger services and other activities.
- Content providers are intermediaries between services and activities.

**Intents**
1. Directed Intents
2. Broadcast Intents

**BIG PICTURE**
Execution of an Android Application

GUI State

Concatenation of the following:

1. Java Package Name,
2. Activity Name,
3. Contextual States (WiFi, Orientation etc.),
4. GUI Components - Their sizes, labels, and accessibility.
Execution of an Android Application

GUI Action

Actions performed by a user: text, click, swipe etc.
## List of GUI Actions

**Tablo: List of all GUI Actions**

<table>
<thead>
<tr>
<th>Non-Contextual</th>
<th>Param1</th>
<th>Param2</th>
<th>Param3</th>
<th>Param4</th>
<th>Param5</th>
</tr>
</thead>
<tbody>
<tr>
<td>click</td>
<td>x</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>longclick</td>
<td>x</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>text</td>
<td>x</td>
<td>y</td>
<td>string</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>swipe</td>
<td>x1</td>
<td>y1</td>
<td>x2</td>
<td>y2</td>
<td>duration</td>
</tr>
<tr>
<td>menu</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>back</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contextual</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectivity</td>
<td>on/off/toggle</td>
</tr>
<tr>
<td>bluetooth</td>
<td>on/off/toggle</td>
</tr>
<tr>
<td>location</td>
<td>gps/gps&amp;network/off/toggle</td>
</tr>
<tr>
<td>planemode</td>
<td>on/off/toggle</td>
</tr>
<tr>
<td>doze</td>
<td>on/off/toggle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Special</th>
<th>Param1</th>
<th>Param2</th>
<th>Param3</th>
<th>Param4</th>
<th>Param5</th>
</tr>
</thead>
<tbody>
<tr>
<td>reinitialize</td>
<td>package</td>
<td>activity</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Proceed to DEMO.
What does Monkey do?

- Randomly generates
  1. **System events** and
  2. **GUI actions**.
- Comes with the Android OS.
- **Very fast**, thousands of actions in a second.
## Monkey Pros/Cons

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Unrealistic input</td>
</tr>
<tr>
<td>Many kinds of actions</td>
<td>Can’t go to deep into the application</td>
</tr>
</tbody>
</table>
Monkey Pros/Cons

Pros
- Speed
- Many kinds of actions

Cons
- Unrealistic input
- Can’t go to deep into the application.
Monkey Pros/Cons

→ click More

**Pros**
- Speed
- Many kinds of actions

**Cons**
- Unrealistic input
- Can’t go to deep into the application.
Monkey Pros/Cons

---

**Pros**
- Speed
- Many kinds of actions

**Cons**
- Unrealistic input
- Can’t go to deep into the application.
Monkey Pros/Cons

Pros
- Speed
- Many kinds of actions

Cons
- Unrealistic input
- Can’t go to deep into the application.
Monkey Pros/Cons

Pros
- Speed
- Many kinds of actions

Cons
- Unrealistic input
- Can’t go to deep into the application.

→ **click** cache view
Monkey Pros/Cons

**Pros**
- Speed
- Many kinds of actions

**Cons**
- Unrealistic input
- Can’t go to deep into the application.

---

Unfortunately, aagtl has stopped.

→ Monkey CAN’T detect the crash. Too deep in the app.
Proceed to DEMO.
Publicly Available Tools

1. **A³E**: Targeted Exploration. Uses **Static Activity Transition Graph (SATG)** to test yet unexplored activities or test the activities that have a transition to unexplored activities.

2. **DynoDroid**: A random tester that gives bias towards relevant events that **trigger relevant methods**.

3. **SwiftHand**: Learns a **finite-transition model** of the Application Under Test (AUT) to **minimize restarts**.

4. **PUMA**: Introduces **cosine similarity** between GUI states.

5. **Sapienz**: Uses **evolutionary algorithms** to generate test cases.

!! None of the tools detect as many crashes as **Monkey**.
# Measures of Testing Tool Performance

## Crashes

- **Number of Crashes Detected**: The main goal of testing is to detect as many crashes as possible.

- **Number of Distinct Crashes**: Testing tools may abuse the same crash for performance increase. Must count each crash once.

## How to compare crashes?

- Get related Android system logs via built-in LogCat.

  - **Assumption**: Similar stack traces correspond to the same crash.
Measures of Testing Tool Performance

What if, there is no crash, or testing tools have the same crash performance?

Then, measure how much of the application is covered.

<table>
<thead>
<tr>
<th>Coverage</th>
<th>High-Level Coverage</th>
<th>Low-Level Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Coverage (used)</td>
<td>Class Coverage (not used)</td>
</tr>
<tr>
<td></td>
<td>Widget Coverage (not used)</td>
<td>Method Coverage (used)</td>
</tr>
<tr>
<td></td>
<td>Event Coverage (not used)</td>
<td>Branch Coverage (used)</td>
</tr>
<tr>
<td></td>
<td>State Coverage (not used)</td>
<td>Statement Coverage (used)</td>
</tr>
</tbody>
</table>
AndroFrame

What is AndroFrame?

- **Fully automated**, 
- **Model learning**, and
- **Black-box**

Features

- Extended Labeled Transition System (ELTS).
- **Action Decisions**: Machine-Learning Based.
**Action:** reinitialize com.tum.yahtzee MainActivity
**Action:** click 200 390 (click play)
Action: click 200 410 (click ok)
**Action:** text 200 270 12345 (text1)
**Action:** reinitialize com.tum.yahtzee MainActivity
AndroFrame Example

**Action:** text 200 270 12345 (text1)
**Action:** text 200 330 12345 (text2)

Diagram:

- Initial state
- Transition: **v1** -> **v2**
- Transition: **v2** -> **v1**
- Transition: **v1** -> **v2**
- Transition: **v2** -> **v1**

States:
- **v1**: reinitialize
- **v2**: click

Restrictions:
- **text1**, **text2**
**Action:** click 200 390 (click play)
**Action:** click 200 390 (click play)
Reinforcement Learning

State and Action Abstractions

\[ \beta(v) = \begin{cases} 
1, & |\lambda(v)| \leq 1 \\
2, & |\lambda(v)| \leq 3 \\
3, & |\lambda(v)| \leq 8 \\
4, & |\lambda(v)| \leq 15 \\
5, & |\lambda(v)| > 15 
\end{cases} \quad \alpha(z) = \begin{cases} 
1, & z \text{ is a menu} \\
2, & z \text{ is a back} \\
3, & z \text{ is a click} \\
4, & z \text{ is a longclick} \\
5, & z \text{ is a text} \\
6, & z \text{ is a swipe} \\
7, & z \text{ is a contextual} 
\end{cases} \]
### Reinforcement Learning

**Q-Matrices as Expectation Distributions for Multiple Objectives**

\[
\vec{Q}_a = \begin{bmatrix}
0.11 & 0.09 & 0.40 & 0 & 0.10 & 0.30 & 0 \\
0.13 & 0.44 & 0.26 & 0 & 0.12 & 0.05 & 0 \\
0.06 & 0.66 & 0.16 & 0 & 0.13 & 0 & 0 \\
0.17 & 0.25 & 0.40 & 0 & 0.09 & 0.09 & 0 \\
0.06 & 0.28 & 0.52 & 0 & 0.09 & 0.05 & 0 
\end{bmatrix}
\]  

(3)

\[
\vec{Q}_c = \begin{bmatrix}
0.04 & 0.18 & 0.33 & 0 & 0.12 & 0.33 & 0 \\
0.19 & 0.18 & 0.12 & 0 & 0.44 & 0.07 & 0 \\
0.13 & 0.43 & 0.15 & 0 & 0.07 & 0.23 & 0 \\
0.17 & 0.18 & 0.48 & 0 & 0.18 & 0 & 0 \\
0.33 & 0.26 & 0.13 & 0 & 0.23 & 0.04 & 0 
\end{bmatrix}
\]  

(4)
Case Study 1: Loop-Stressing

Pressing **Coin** button multiple times results in crash.
Case Study 2: Contextual-State Toggling

Turning **bluetooth** on and then clicking **Find Devices** results in crash.
Case Study 3: Pause-Resume

Pausing and then resuming results in a crash.
Test Case Mutation (TCM)

Case Study 4: Change Text

Changing text results in a crash.
Test Case Mutation (TCM) Example

(a) Test Cases
(b) AUT Model
(c) Minimization and Mutation

Fig. 2: Motivating Example (mutations are denoted as bold)
Some Results

Experimental Set

100 Applications from known F-Droid benchmarks.
## Future Work

### App-Agnostic Oracles

Automated oracles that **find non-crashing problems** in Android.

- Pausing-Resuming not returning the same state.
- Broken layout after double rotation.
- Broken back button not going to previous state.

### Feedback-Directed Monkey (FDMonkey) Testing

- Monkey can’t go **deep into the application**.
- **Guide Monkey parameters** using the coverage, crash and other info.
Specification-Based Testing

It is not interesting to test some applications for crash.
Specification-Based Testing

Test for the output correctness via specifications
Thank You. Any Questions?