An Infrastructure Approach to Improving Effectiveness of Android UI Testing Tools

Wenyu Wang, Wing Lam, Tao Xie

{wenyu2, winglam2}@illinois.edu
taoxie@pku.edu.cn
Automated UI Testing For Android Apps

Automatically explore the app through UIs, just like human users

- Code coverage
- # of crashes
- Test effectiveness report
- Capturing of screen content
- Test device
- Action(s) (screen tap, key press, ..)
- Test tool
- Test record and report

👍 Little human effort
👍 Scalable with numerous devices
👍 Deeper functionality saturation

How much do these two types of operations affect testing effectiveness?
Infrastructure Efficiency: A Motivational Study

• Break down of testing time usages
  • 3 tools using UiAutomator from the 2018 study \(^1\)
    • Including one re-implemented Monkey (baseline tool), Chimp
  • 15 industrial apps from the study, each run for 1-hour
    • 1m+ to 1b+ downloads, 3.3MB to 93MB APK sizes

\[\text{UI Capturing: 34\%} \]
\[\text{UI Event Execution: 36\%} \]
\[\text{Misc Interaction: 9\%} \]
\[\text{Tool Internal: 21\%} \]

70\% of testing time is spent on just two types of operations!

Save time from these operations →
More actions within limited testing time →
Better testing effectiveness!

\(^1\) Wenyu Wang, Dengfeng Li, Wei Yang, Yurui Cao, Zhenwen Zhang, Yuetang Deng, and Tao Xie.  
An empirical study of android test generation tools in industrial cases (ASE 2018)
Our Approach (Toller)

- **Goal**: Fast UI Hierarchy Capturing + UI Event Execution
- **Direct access** to app UI data structures & event handlers
- Low-overhead communication with in-app agent
UI Hierarchy Capturing

Obtain structured on-screen contents from the test device
Mechanism Of UI Hierarchy Capturing (UIAutomator)
Mechanism Of UI Hierarchy Capturing (Toller)

Toller facilitates direct access to UI data structures!
UI Event Execution

TOLLER facilitates direct access to event handlers!

High-level event
e.g., tapping button X

Low-level event
tapping \((x,y)\) on screen

Event Handler
X's OnClickListener

App VM

Android UI System
- Hardware Composer
- Surface Flinger
- Window Manager

UI Hierarchy Capturing
Test device
Test tool
Evaluation Outline

• RQ1: Efficiency of two types of operations
• RQ2: Code coverage improvement
• RQ3: Crash triggering ability improvement
• RQ4: Code/crash overlap with and without Toller
  • Please see paper[1] for details
• RQ5: Breakdown of improvements by enhancing types of two operations
  • Please see paper[1] for details

RQ1: Efficiency Comparison

- Same testing time, with and without **Toller**
- Fallback to UIAutomator on unhandled cases

**Toller substantially accelerates two types of operations**
RQ2: Code Coverage Improvement

- 3 one-hour runs for each (tool, app)
- Average # of Java methods covered after testing starts

+11.8%, 10.4%, 70.1% on CH, WT, and ST

#apps with highest coverage:

<table>
<thead>
<tr>
<th></th>
<th>MK</th>
<th>CH</th>
<th>WT</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Toller</td>
<td>4</td>
<td>8</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>With Toller</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Toller-induced coverage improvements are substantial enough to change relative tool competitiveness

MK = Monkey, CH = Chimp (re-implemented Monkey), WT = WCTester, ST = Stoat
RQ2: Code Coverage Improvement

- Additionally evaluate on Ape
  - More advanced algorithm than tools from the 2018 study
  - No mention of leveraging private APIs for UI Hierarchy Capturing
  - Slow (no efficient infra. support) vs. Original (with infra. support)

Tools with less advanced algorithm but efficient infra. support could outperform tools with more advanced algorithm but no efficient infra. support

Ape_S lower than WCTester_E for ~40 minutes

9.7% improvement from Ape_S to Ape_O

10.4% improvement from WCTester_O to WCTester_E
RQ3: Crash Triggering Improvement

- Cumulative # of distinct crashes, identified by stacktraces
- 3.6x, 1.5x, 1.4x for three enhanced tools; 1.8x for Ape
- For the majority of (tool, app) pairs, more crashes are found by enhanced tool versions

MK = Monkey, CH = Chimp (re-implemented Monkey), WT = WCTester, ST = Stoat

Efficient infrastructure helps tools trigger substantially more crashes

<table>
<thead>
<tr>
<th>App Name</th>
<th>APE_S</th>
<th>%S</th>
<th>APE_O</th>
<th>%O</th>
<th>∑APE</th>
<th>MK</th>
<th>CH_O</th>
<th>%O</th>
<th>CH_E</th>
<th>%E</th>
<th>∑CH</th>
<th>WT_O</th>
<th>%O</th>
<th>WT_E</th>
<th>%E</th>
<th>∑WT</th>
<th>ST_O</th>
<th>%O</th>
<th>ST_E</th>
<th>%E</th>
<th>∑St</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>1</td>
<td>33%</td>
<td>2</td>
<td>67%</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Duolingo</td>
<td>1</td>
<td>50%</td>
<td>1</td>
<td>50%</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Filters For Selfie</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>100%</td>
<td>5</td>
<td>100%</td>
<td>5</td>
<td>13%</td>
<td>7</td>
<td>88%</td>
<td>8</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GoodRx</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Google Translate</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Marvel Comics</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Merriam-Webster</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mirror</td>
<td>3</td>
<td>60%</td>
<td>5</td>
<td>100%</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>60%</td>
<td>5</td>
<td>100%</td>
<td>5</td>
<td>83%</td>
<td>4</td>
<td>67%</td>
<td>6</td>
<td>5</td>
<td>63%</td>
<td>7</td>
<td>88%</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>My Baby Piano</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sketch</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>trivago</td>
<td>1</td>
<td>33%</td>
<td>2</td>
<td>67%</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WEBTOON</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Word</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>100%</td>
<td>4</td>
<td>100%</td>
<td>4</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>100%</td>
<td>3</td>
<td>100%</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>Youtube</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>3</td>
<td>100%</td>
<td>3</td>
<td>1</td>
<td>100%</td>
<td>3</td>
<td>100%</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>Zedge</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>1</td>
<td>100%</td>
<td>3</td>
<td>100%</td>
<td>3</td>
<td>1</td>
<td>100%</td>
<td>3</td>
<td>100%</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>8</td>
<td>42%</td>
<td>14</td>
<td>74%</td>
<td>19</td>
<td>15</td>
<td>5</td>
<td>25%</td>
<td>18</td>
<td>90%</td>
<td>20</td>
<td>43%</td>
<td>20</td>
<td>67%</td>
<td>30</td>
<td>92</td>
<td>61%</td>
<td>125</td>
<td>82%</td>
<td>152</td>
<td>-</td>
</tr>
</tbody>
</table>
Recap & Conclusion

• Over 70% of testing time budget is for Android testing tools' use of test infrastructure

• Use of test infrastructure can be made much more efficient with TOLLER
  • 10.4% - 70.1% code coverage improvement, 1.4x - 3.6x unique crashes detected depending on tool

• Code and data available at https://github.com/TOLLER-Android/main

Efficient infrastructure support is useful for effective Android UI testing tools, complementary with existing algorithmic advances