Generating various contexts from permissions for testing Android applications

Kwangsik Song, Ah-Rim Han, Sehun Jeong, Sungdeok Cha

Presented by Ah-Rim Han

Korea University, South Korea

2015. 7. 6

Contents

- Introduction
- Overview
- Testing Android applications in various contexts
- Evaluation
- Conclusion and future Work

* http://www.peakvision.org

Testing for mobile applications

→ Example of context-aware application: Peak Vision*
- Medical images can be captured using a clip-on camera adapter
- Images can be sent to the systems to perform diagnosis remotely

- Mobile application is context-aware application
  - Can provide rich, context-aware contents to users
  - Designed to be aware of the computing context in which it runs and adapt and react according to its findings
  - Should be testable in any environment and in any contextual inputs

→ Need to test applications considering complex, various contexts

Motivation

- The existing studies have limitations
  - GUI testing : Monkey (Random testing) [1] and Android GUI Ripper (Model-based testing) [2]
    - Focus on GUI events

    → Difficult to find failures that could be detected by considering the changes in the contexts

  - Context-aware testing : Amalfitano’s work [3]
    - Specific event sequences generated based on a limited number of scenarios (event patterns) were considered

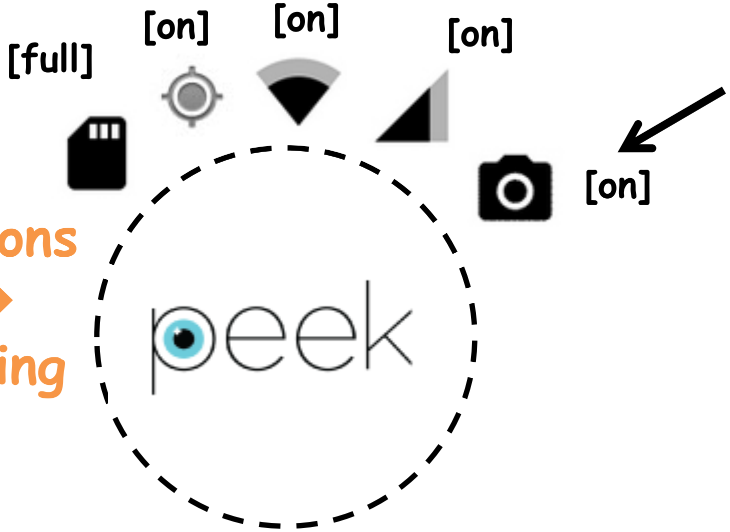
    → Difficult to find bugs that occur in various complex contexts

→ We need a systematic method for generating executing contexts

Our idea

- We can easily infer the related resources using *permissions*
  - Android application includes permissions (in a manifest file)
  - *By varying conditions of resources, we can simulate the changing external environment*
    - *States of resource conditions are changed via events*

- To test Android applications, we use permissions to generate the various executing contexts



AndroidManifest.xml
- Permission lists

Permuting resource conditions having variable states → Generating various executing contexts

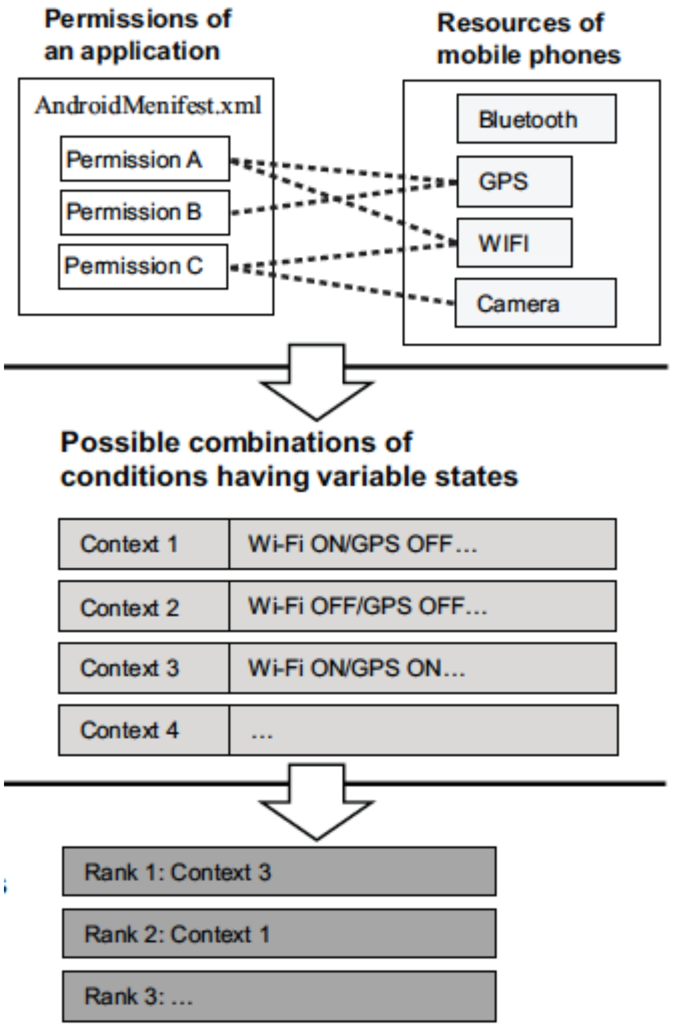
Goal of our approach

- We provide a method for generating various executing contexts from permissions
  - Identifying related resource from permissions
  - Generating various executing contexts
  - Prioritizing contexts by two-level strategies

→ Issue: executing contexts should be prioritized because there are too many executing contexts

Overview

‹ An overview for generating various contexts after analyzing permissions ›



Permissions of an application

AndroidMenifest.xml

- Permission A
- Permission B
- Pemission C

Resources of mobile phones

- Bluetooth
- GPS
- WIFI
- Camera

Possible combinations of conditions having variable states

Context 1	Wi-Fi ON/GPS OFF…
Context 2	Wi-Fi OFF/GPS OFF…
Context 3	Wi-Fi ON/GPS ON…
Context 4	…

- Rank 1: Context 3
- Rank 2: Context 1
- Rank 3: …

- Identifying related resources from permissions
- Generating various executing contexts
- Prioritizing executing contexts

Identifying related resources from permissions

- Identifying resources from permissions
- Defining possible states for each resource

Permission	Allows an app to	Related Resources [Possible States]
ACCESS_FINE_LOCATION	Access precise location from location resources	Wi-Fi[on|off] GPS[on|off] Radio[on|off]

* ACCESS_FINE_LOCATION
→ permission for acquiring right to access detail position

Generating various executing contexts

- Executing contexts can be generated by permuting resource conditions having variable states

Resource 1 [State1|State2] x Resource2 [State3|State4] x Resource 3 [State5|State6]

= 2^3 = 8 (the total number of generated executing contexts)

Wi-Fi	Radio	GPS
on	on	on
on	off	on
on	off	off
on	on	off
……	…..	…..

Prioritizing contexts

- Prioritizing strategies
  - 1) Weighting each resource condition according to the testing objectives
  - 2) Weighting individual or combinatorial resources residing in an executing context.

	Rank	Wi-Fi	GPS	Radio	SD Card	Camera
1)	1	on	on	on	free	enable
	2	off	off	off	full	disable
2)	3	on	on	on	full	enable
	4	off	on	off	free	enable

1)

Normal scenario

Active

Wi-Fi, GPS, Radio=on, SD card = free, Camera = enable

Exceptional scenario

Inactive

Wi-Fi, GPS, Radio=off, SD card = full, Camera = disable

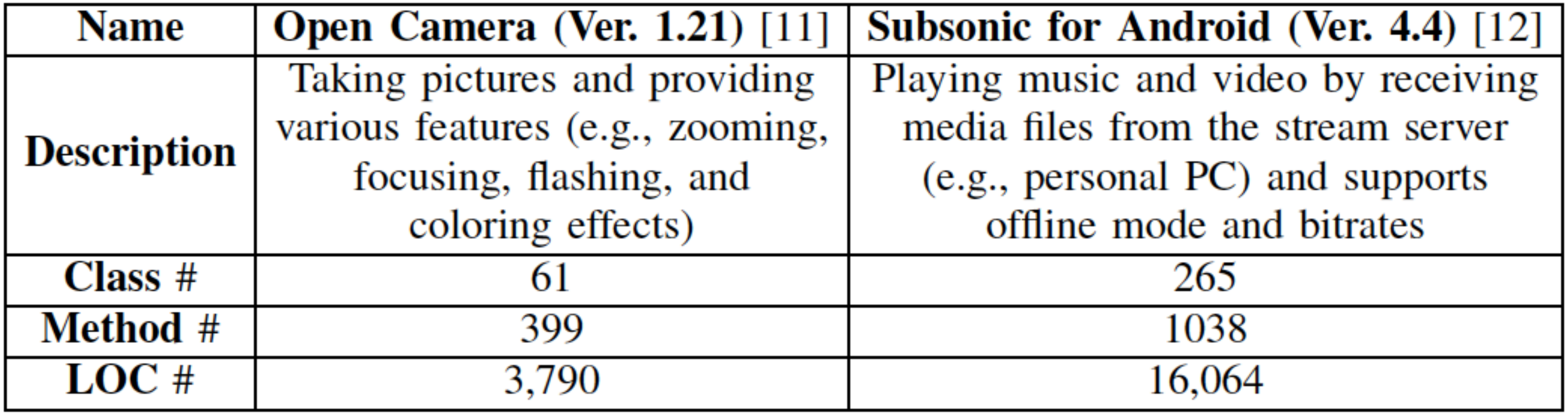
2)

Scenarios capturing fault behavior

- SD card = full
- Wi-Fi = off, GPS = on

Evaluation

- Experimental design

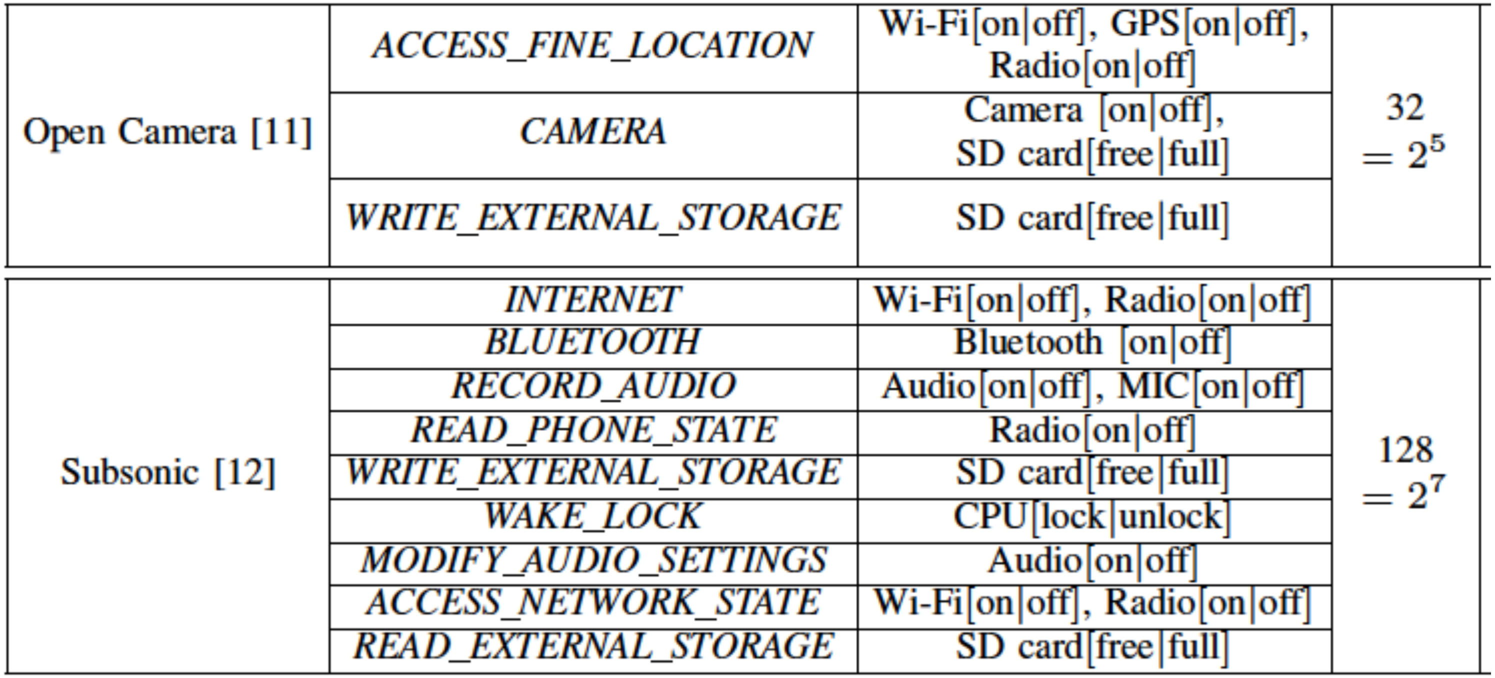


Name	Open Camera (Ver. 1.21) [11]	Subsonic for Android (Ver. 4.4) [12]
Description	Taking pictures and providing various features (e.g., zooming, focusing, flashing, and coloring effects)	Playing music and video by receiving media files from the stream server (e.g., personal PC) and supports offline mode and bitrates
Class #	61	265
Method #	399	1038
LOC #	3,790	16,064

→ Are open source projects
→ Have development histories (e.g., bug issues)
→ Contain large number of classes and methods

Experimental design (1/2)

- Under each contexts, test cases (TCs) are executed
  - TCs are generated from the Android GUI ripper tool [2]
  - TC are also extracted manually
    - Focusing on scenarios used more frequently and faulty behavior may be more occurred



Subject	Permission	Resources[States]	Total#
Open Camera [11]	*ACCESS_FINE_LOCATION*	Wi-Fi[on|off], GPS[on|off], Radio[on|off]	32 $= 2^5$
	CAMERA	Camera [on|off], SD card[free|full]	
	WRITE_EXTERNAL_STORAGE	SD card[free|full]	
Subsonic [12]	*INTERNET*	Wi-Fi[on|off], Radio[on|off]	128 $= 2^7$
	BLUETOOTH	Bluetooth [on|off]	
	RECORD_AUDIO	Audio[on|off], MIC[on|off]	
	READ_PHONE_STATE	Radio[on|off]	
	WRITE_EXTERNAL_STORAGE	SD card[free|full]	
	WAKE_LOCK	CPU[lock|unlock]	
	MODIFY_AUDIO_SETTINGS	Audio[on|off]	
	ACCESS_NETWORK_STATE	Wi-Fi[on|off], Radio[on|off]	
	READ_EXTERNAL_STORAGE	SD card[free|full]	

Experimental design (2/2)

- Research questions
  1) ***RQ 1.*** Is our testing approach useful for detecting faults?

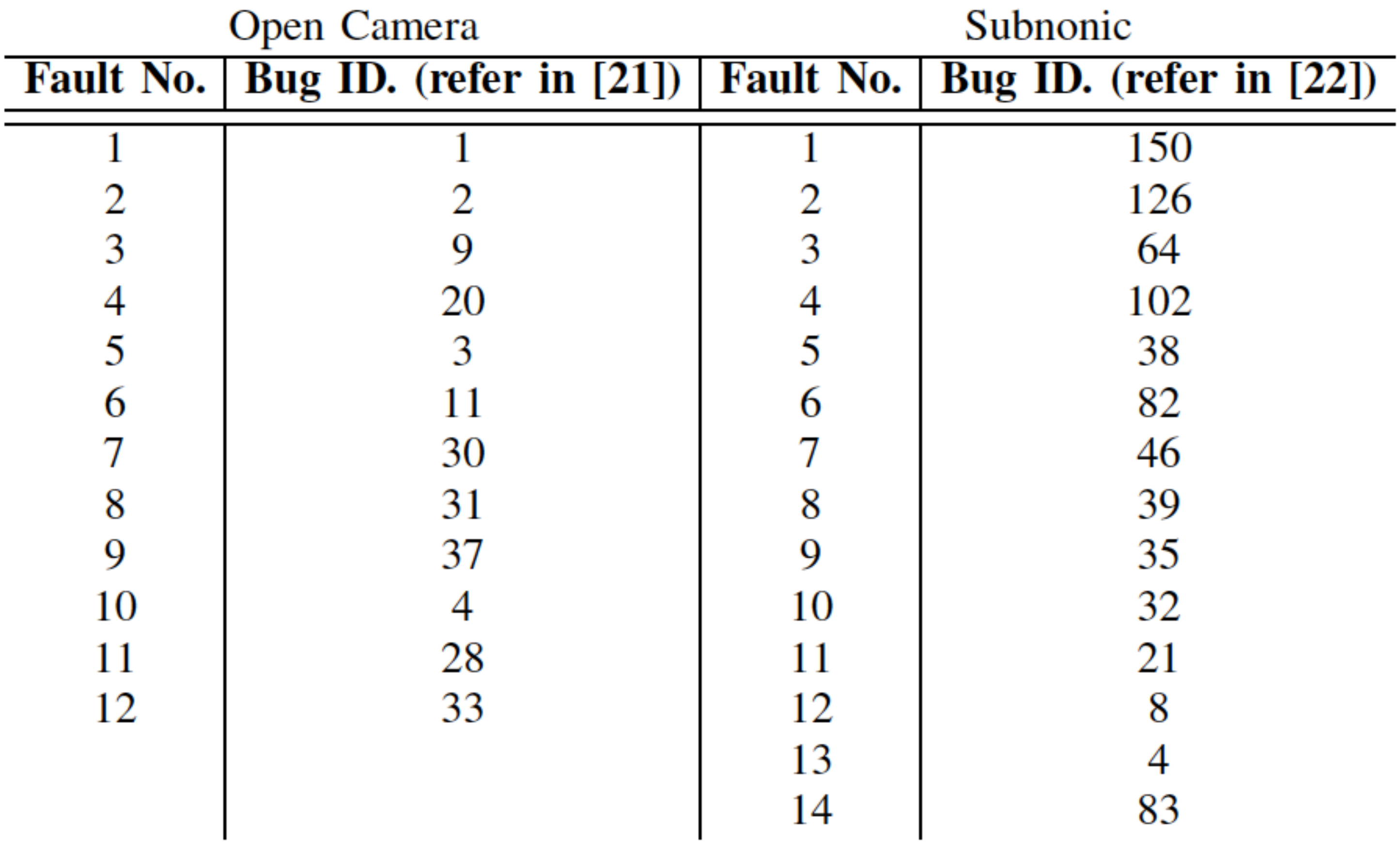
     → Number of detected bugs

  2) ***RQ 2.*** Is our prioritization technique effective in detecting faults?

     → APFD (Average Percentage of Fault Detection)

     → Fault detection rate

Results: Number of detected bugs



Open Camera		Subnonic	
Fault No.	**Bug ID. (refer in [21])**	**Fault No.**	**Bug ID. (refer in [22])**
1	1	1	150
2	2	2	126
3	9	3	64
4	20	4	102
5	3	5	38
6	11	6	82
7	30	7	46
8	31	8	39
9	37	9	35
10	4	10	32
11	28	11	21
12	33	12	8
		13	4
		14	83

→ (# detected bugs) / (# faults existing in the repository)

→ Open Camera : 12 / 38, Subsonic: 14 / 151

Results: APFD measure

T (generated order) , Tr (reversed order), Tp (Prioritized order using our approach)

- Open Camera

(TC 32, Fault 12)

Order	result
T	0.92
Tr	0.62
Tp	0.97

1)

- Subsonic

(TC 128, Fault 14)

Order	result
T	0.96
Tr	0.92
Tp	0.98

2)

1) In both projects, the APFDs for Tp represent the highest scores.

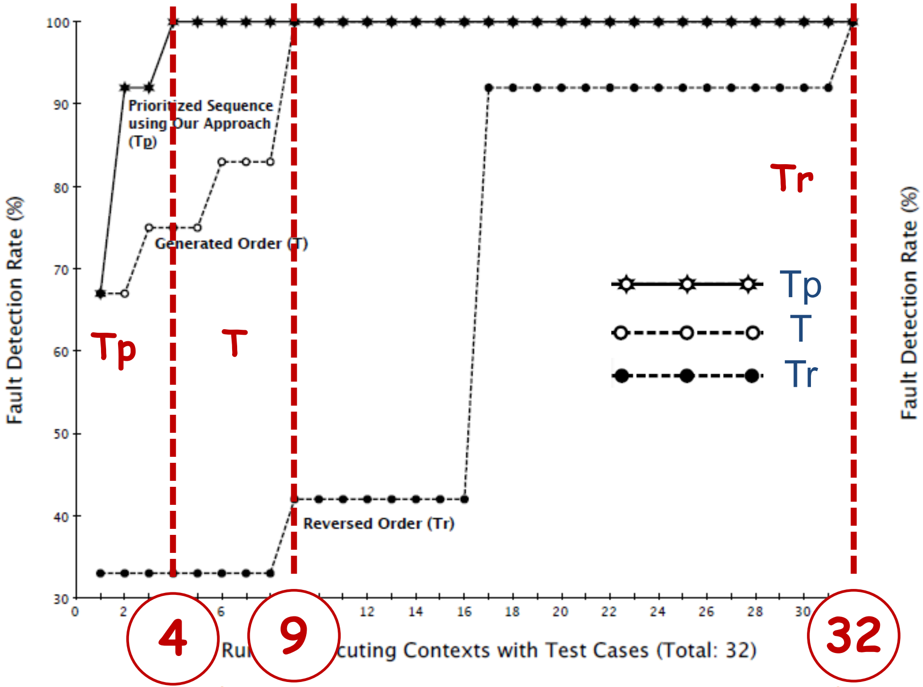
2) In Subsonic, the APFDs are not much different in three orders

: Many of the faults are detected by small number of executing contexts

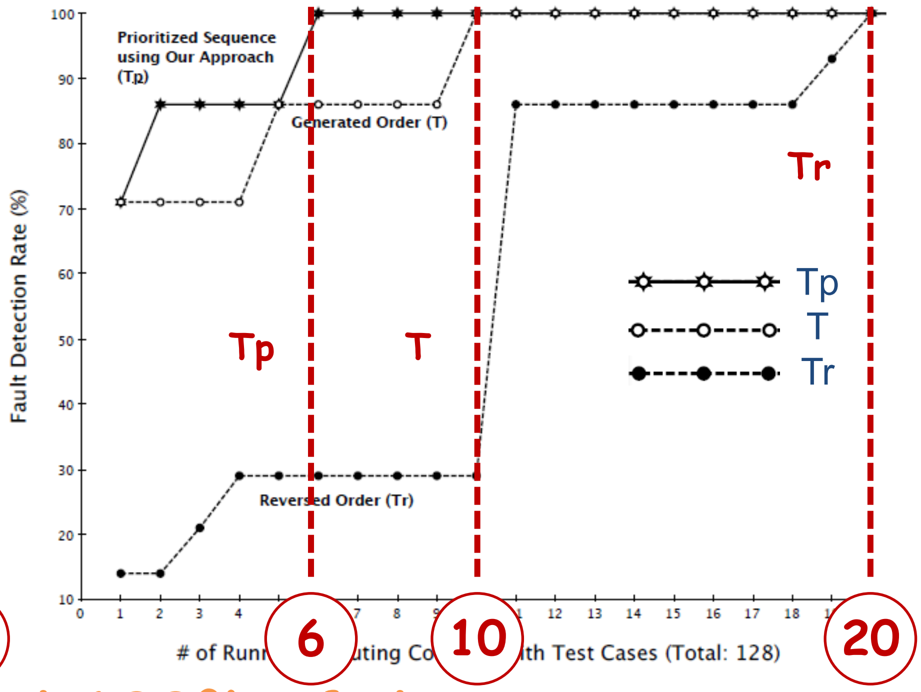
Results: Fault detection rate

T (generated order) , Tr (reversed order), Tp (Prioritized order using our approach)

- Open Camera



- Subsonic



In both projects, Tp reached 100% of detection rate faster by running smaller # of executing contexts

∴ The prioritized order results in the earliest detection of the faults

Conclusion and future Work

- Summary
  - Proposes an efficient method for generating various executing contexts
- Future Work
  - Performs the more detailed experiment
  - Devises the method of considering sequences in our contexts for simulating dynamically changing environment

References

- [1] Monkey. DOI= http://developer.android.com/tools/help/monkey.html
- [2] D. Amalfitano, A.R. Fasolino, P.Tramontana, S. DeCarmine, and A. M. Memon. Using GUI ripping for automated testing of Android applications. Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 2012), 2012. ACM, pp. 258-261
- [3] Domenico Amlfitano, Anna Rita Fasolino, Porfirio Tramontana, and Nicola Amatucci, "Considering Con-text Events in Event-Based Testing of Mobile Applications" IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops, 2013
- [4] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Permission evolution in the android ecosystem," in Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012, pp. 31–40.

Thank you.