UFMG

lab-soft
software engineering laboratory

Resource Interactions in Mobile Applications: Three Presentations for the SPL Testing Community

Euler Marinho

Eduardo Figueiredo (Advisor)

Fischer Ferreira (Coadvisor)

SPLC 2024
28th ACM International Systems and Software Product Line Conference
Luxembourg

Software Engineering Lab (LabSoft)
http://labsoft.dcc.ufmg.br/

Presentation Overview

- Characterizing Resource Interaction Failures in Mobile Applications (Doctoral Symposium)
- Resource Interaction Failures in Mobile Applications: A Challenge for the Software Product Line Testing Community (Challenges and Solutions Track)
- RIFDiscoverer: A Tool for Finding Resource Interaction Failures (Demonstrations and Tools Track)

SPLC 2024 – Doctoral Symposium

CHARACTERIZING RESOURCE INTERACTION FAILURES IN MOBILE APPLICATIONS

Summary

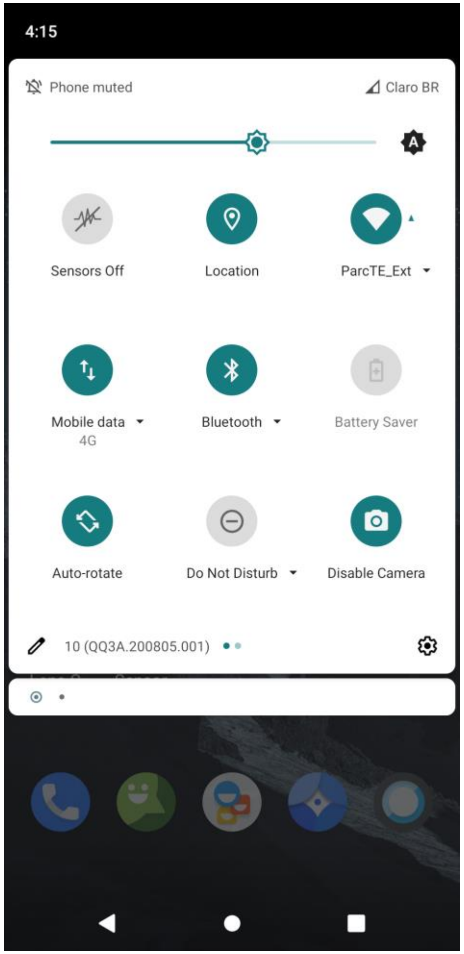
- Introduction
- Main Research Questions
- Research Method
- Preliminary Results
- Work Plan

Introduction

- Mobile devices have a rich set of resources
- “Resource” refers to sensors, radios, and user-controlled options
- User interaction with devices can enable or disable the resources
- Unexpected application behavior can occur in specific resource settings
- However, the testing of all input combinations is impracticable

Resources in mobile applications

- Platform configurations
  - Enabled/disabled resources
- Communication features
  - Wi-Fi, Bluetooth, etc
- Sensors
  - Accelerometer, Gyroscope, etc
- User-controlled options
  - Battery saver, Auto-rotate, etc



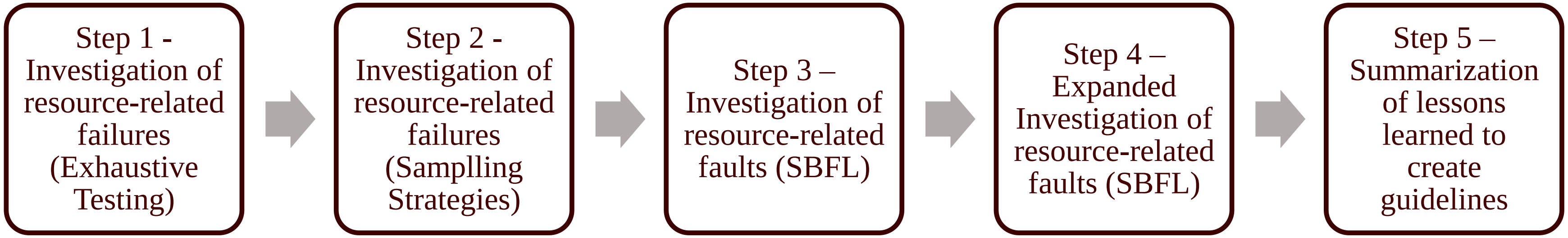
Sampling Strategies

- Resource interactions are like Feature interactions
- Resource settings are 14-tuple of resource and state pairs
- Sampling strategies are alternatives for decreasing the testing effort
- Random, One Enabled, One Disabled, Most Enabled Disabled, Pairwise

Main Research Questions

- RQ1: Which resource interactions more frequently cause failures?
- RQ2: Which sampling strategies are the most effective to find resource interaction failures in mobile applications?
- RQ3: To what extent the Spectrum-based Fault Localization technique can be used for locating faults in mobile applications?

Research Method



Step 1 - Investigation of resource-related failures (Exhaustive Testing)

Step 2 - Investigation of resource-related failures (Samplling Strategies)

Step 3 – Investigation of resource-related faults (SBFL)

Step 4 – Expanded Investigation of resource-related faults (SBFL)

Step 5 – Summarization of lessons learned to create guidelines

Dataset

- 20 Android applications
- 14 target resources
  - Auto Rotate, Battery Saver, Bluetooth, Camera, Do Not Disturb, Location, Mobile Data, Wi-Fi, Accelerometer, Gyroscope, Light, Magnetometer, Orientation, Proximity
- Extended test suites

Dataset Excerpt

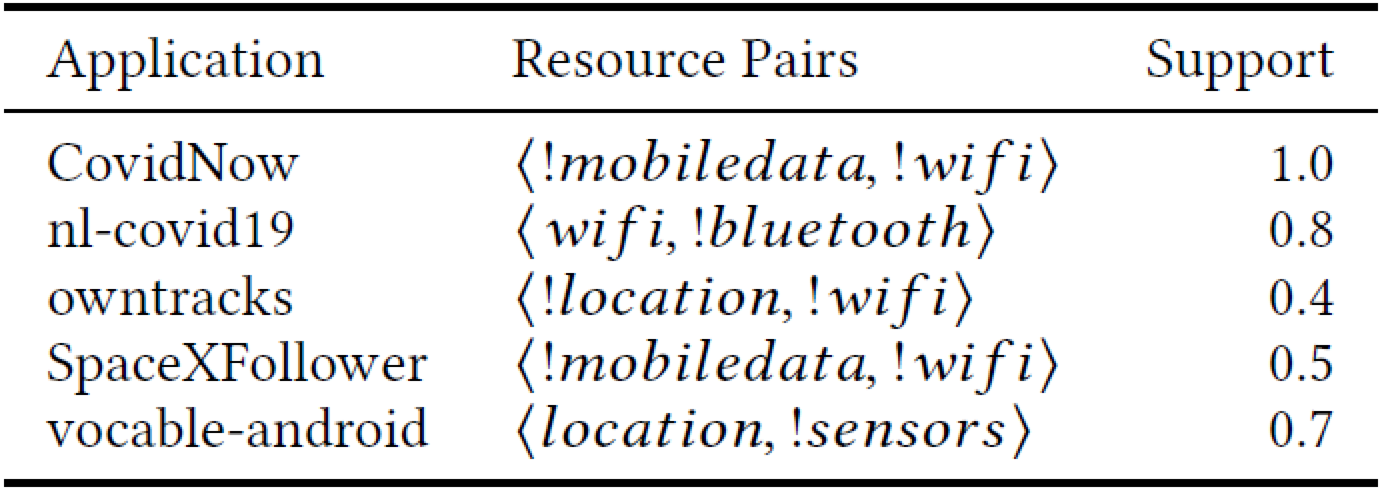
NAME	LOC	#Test Cases	Test LOC	RESOURCES
AnkiDroid	158 K	164	2,770	Cam, MD, Wi-Fi
CovidNow	2 K	21	540	MD, Wi-Fi
Iosched	27 K	9	473	Loc, MD, Wi-Fi
Mixin-Messenger	168 K	160	3,732	BT, Cam, Loc, MD, Wi-Fi
Moonshot	**0,455 K**	28	**464**	MD, Wi-Fi
Radio-Droid	22 K	23	1,735	BT, MD, Wi-Fi
WordPress	**347 K**	115	**3,674**	Cam, MD, Wi-Fi

BT - Bluetooth
Cam - Camera
Loc - Location
MD - Mobile Data

Test suite instrumentation

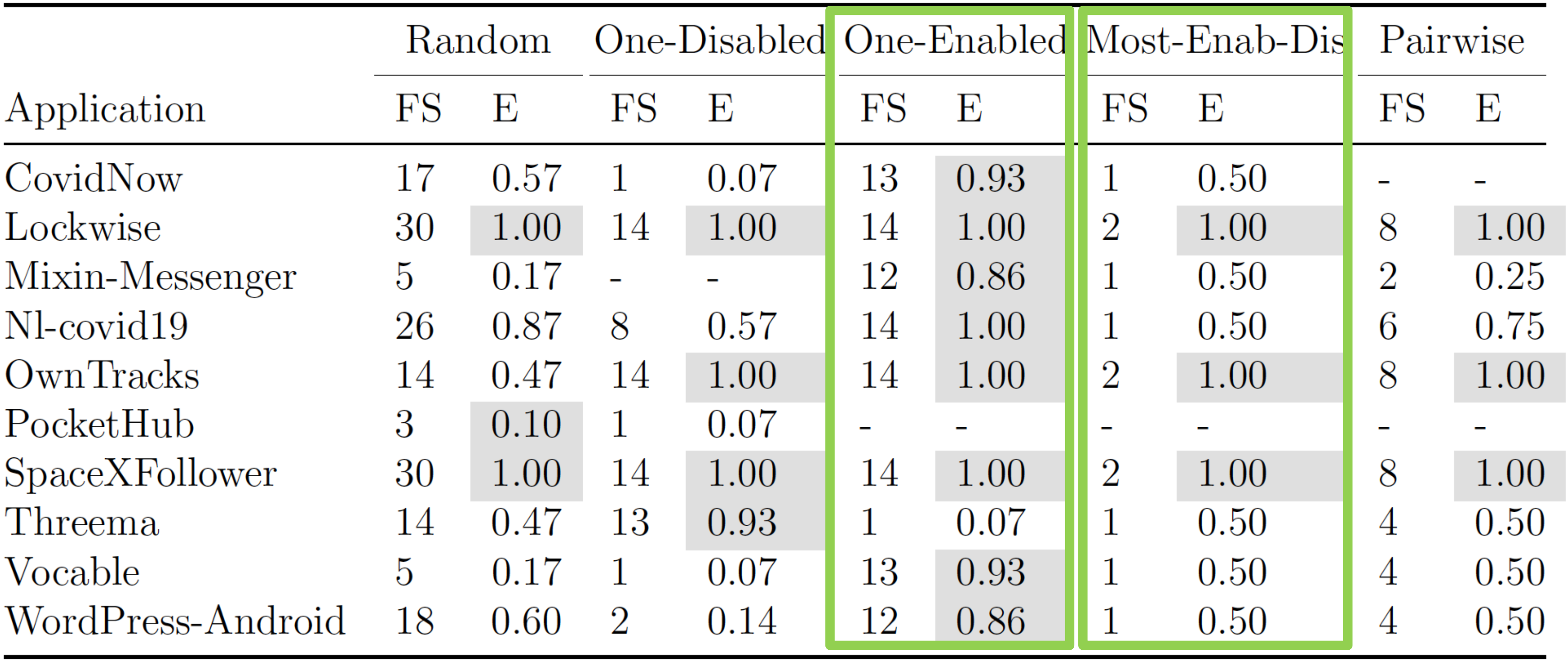
- Functional tests are the target
  - Android APIs for interacting with the device
- Extension by means of UI Automator
- Each test class is extended with instrumentation code
- Before each test case the instrumentation code is executed
- Test reports are processed

Resource Interactions Most Likely to Cause a Failure (RQ1)



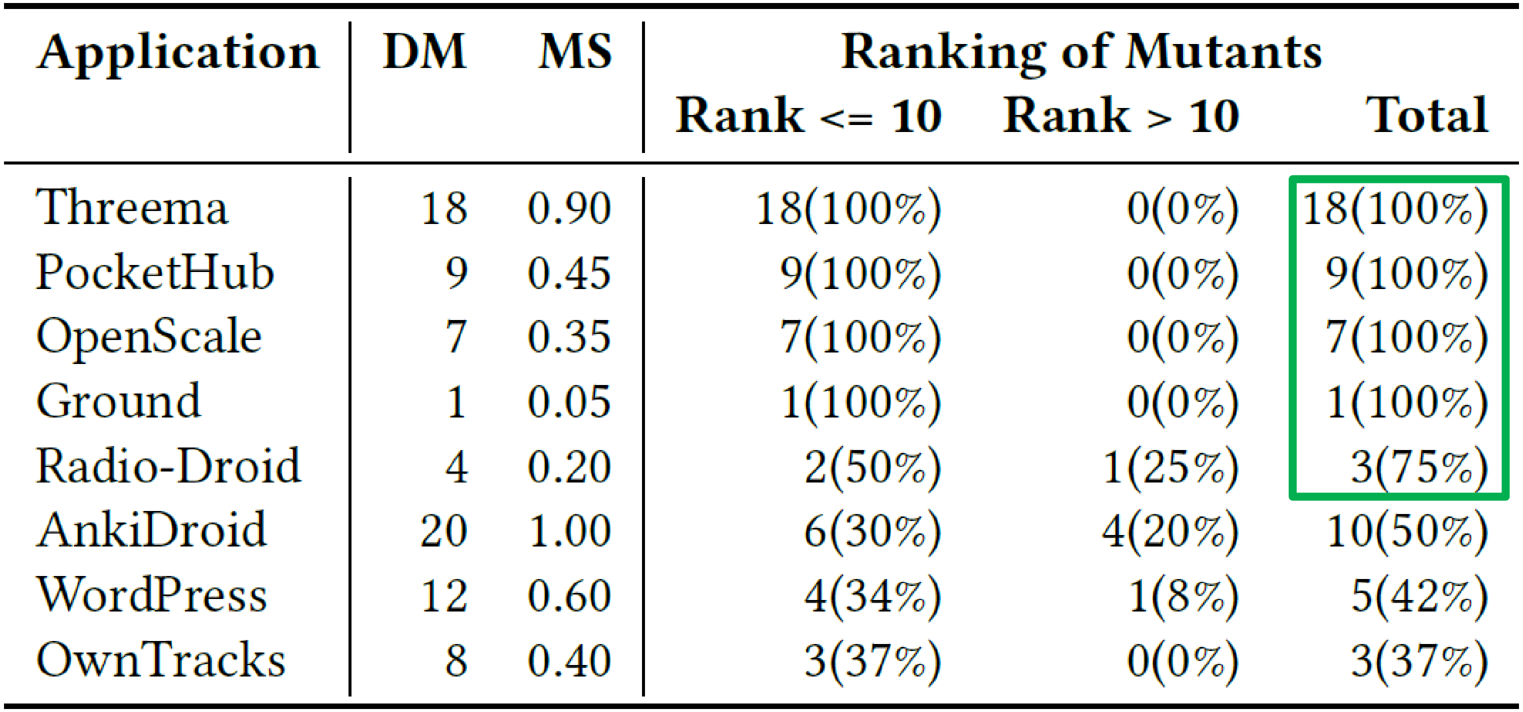
Application	Resource Pairs	Support
CovidNow	$\langle !mobiledata, !wifi \rangle$	1.0
nl-covid19	$\langle wifi, !bluetooth \rangle$	0.8
owntracks	$\langle !location, !wifi \rangle$	0.4
SpaceXFollower	$\langle !mobiledata, !wifi \rangle$	0.5
vocable-android	$\langle location, !sensors \rangle$	0.7

The Most Effective Testing (RQ2)



	Random		One-Disabled		One-Enabled		Most-Enab-Dis		Pairwise	
Application	FS	E	FS	E	FS	E	FS	E	FS	E
CovidNow	17	0.57	1	0.07	13	0.93	1	0.50	-	-
Lockwise	30	1.00	14	1.00	14	1.00	2	1.00	8	1.00
Mixin-Messenger	5	0.17	-	-	12	0.86	1	0.50	2	0.25
Nl-covid19	26	0.87	8	0.57	14	1.00	1	0.50	6	0.75
OwnTracks	14	0.47	14	1.00	14	1.00	2	1.00	8	1.00
PocketHub	3	0.10	1	0.07	-	-	-	-	-	-
SpaceXFollower	30	1.00	14	1.00	14	1.00	2	1.00	8	1.00
Threema	14	0.47	13	0.93	1	0.07	1	0.50	4	0.50
Vocable	5	0.17	1	0.07	13	0.93	1	0.50	4	0.50
WordPress-Android	18	0.60	2	0.14	12	0.86	1	0.50	4	0.50

Use of SBFL for Mobile Applications (RQ3)



Application	DM	MS	Ranking of Mutants		
			Rank <= 10	Rank > 10	Total
Threema	18	0.90	18(100%)	0(0%)	18(100%)
PocketHub	9	0.45	9(100%)	0(0%)	9(100%)
OpenScale	7	0.35	7(100%)	0(0%)	7(100%)
Ground	1	0.05	1(100%)	0(0%)	1(100%)
Radio-Droid	4	0.20	2(50%)	1(25%)	3(75%)
AnkiDroid	20	1.00	6(30%)	4(20%)	10(50%)
WordPress	12	0.60	4(34%)	1(8%)	5(42%)
OwnTracks	8	0.40	3(37%)	0(0%)	3(37%)

* DM = Dead mutants

* MS = Mutation score

Work Plan

- Characterize faults behind the failures
  - How to identify faulty classes?
    - Failures are related to the test framework scope
    - Android event-driven nature is a challenge for debugging activities
- Expanded investigation of Spectrum based Fault Localization
  - Bug fix patterns

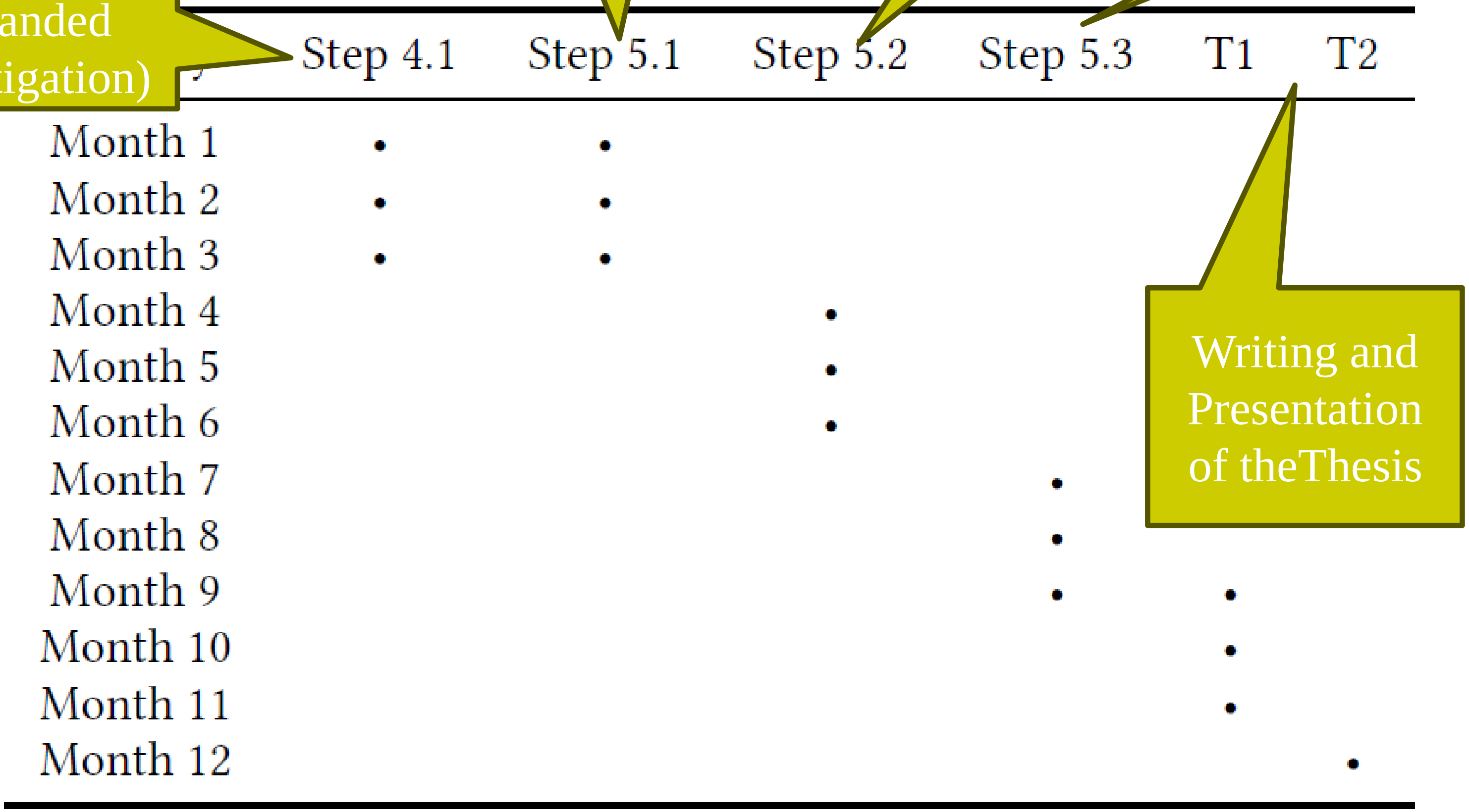
Remaining A s

Writing a Paper (SBFL expanded investigation)

Documentation of Guidelines

Validation of Guidelines

Writing Papers about Guidelines



	Step 4.1	Step 5.1	Step 5.2	Step 5.3	T1	T2
Month 1	●	●				
Month 2	●	●				
Month 3	●	●				
Month 4			●			
Month 5			●			
Month 6			●			
Month 7				●		
Month 8				●		
Month 9				●	●	
Month 10					●	
Month 11					●	
Month 12						●

Writing and Presentation of theThesis

Questions?

SPLC 2024 – Challenges and Solutions Track

CHARACTERIZING RESOURCE INTERACTION FAILURES IN MOBILE APPLICATIONS

Summary

- Introduction
- Dataset Overview
  - Artifacts
  - Test suite instrumentation
  - Failure Reports
  - Example of Use

Introduction

- Mobile devices have a rich set of resources
- “Resource” refers to sensors, radios, and user-controlled options
- User interaction with devices can enable or disable the resources
- Unexpected application behavior can occur in specific resource settings
- However, the testing of all input combinations is impracticable

Sampling Strategies

- Resource interactions are like Feature interactions
- Resource settings are 14-tuple of resource and state pairs
- Sampling strategies are alternatives for decreasing the testing effort
- Random (30), One Enabled (14), One Disabled (14), Most Enabled Disabled (2), Pairwise (8)

Proposed Challenge

- SPLC participants must propose testing strategies for mobile applications
  - Taking resource interactions into account
- The failure detection capability and the effectiveness must be higher than our baseline
  - Increase the number of unique detected failures and minimize the number of tested settings
- Solution efficiency (SE)

$$SE = \frac{FailingSettings}{TotalSettings}$$

Dataset Overview

- 20 Android applications
- 14 target resources
  - Auto Rotate, Battery Saver, Bluetooth, Camera, Do Not Disturb, Location, Mobile Data, Wi-Fi, Accelerometer, Gyroscope, Light, Magnetometer, Orientation, Proximity
- Extended test suites

Dataset Excerpt

NAME	LOC	#Test Cases	Test LOC	RESOURCES
AnkiDroid	158 K	164	2,770	Cam, MD, Wi-Fi
CovidNow	2 K	21	540	MD, Wi-Fi
Iosched	27 K	9	473	Loc, MD, Wi-Fi
Mixin-Messenger	168 K	160	3,732	BT, Cam, Loc, MD, Wi-Fi
Moonshot	**0,455 K**	28	**464**	MD, Wi-Fi
Radio-Droid	22 K	23	1,735	BT, MD, Wi-Fi
WordPress	**347 K**	115	**3,674**	Cam, MD, Wi-Fi

BT - Bluetooth
Cam - Camera
Loc - Location
MD - Mobile Data

Dataset Artifacts

- Application source code and test suites
- Found Failures (CSV)
- Analyzed Settings (CSV)

Test suite instrumentation

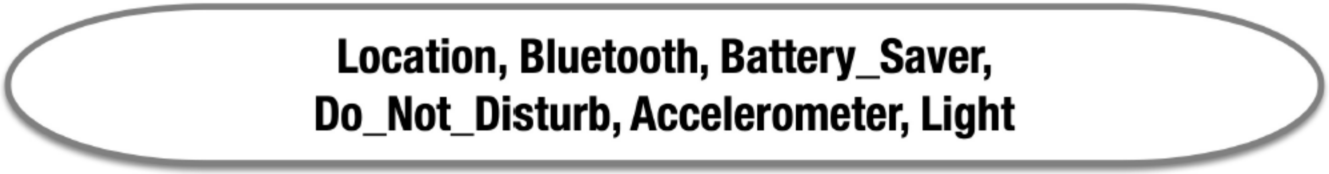
- Functional tests are the target
  - Android APIs for interacting with the device
- Extension by means of UI Automator
- Each test class is extended with instrumentation code
- Before each test case the instrumentation code is executed
- Test reports are processed

Failure Reports

NAME	FAILING SETTINGS	SOLUTION EFFICIENCY	#FAILURES
CovidNow	32	0.47	2
Lockwise	68	1.00	4
Mixin-Messenger	20	0.29	2
Nl-covid19	55	0.81	6
OwnTracks	68	1.00	3
PocketHub	4	0.06	1
SpaceXFollower	68	1.00	4
Threema	33	0.48	1
Vocable	24	0.35	7
WordPress	37	0.54	11

Example of Use

- Settings are provided in CSV files with only enabled resources



Location, Bluetooth, Battery_Saver,
Do_Not_Disturb, Accelerometer, Light

Example of Output

- Vocable is a communication tool for individuals who are speech impaired
- It uses the ARCore SDK to track the user's head movements
  - To understand where the user is looking on the screen
- When both Mobile Data and Wi-Fi are disabled (*verifyDefaultTextAppears* test)
  - ARCore fatal exception

Questions?

SPLC 2024 – Demonstrations and Tools Track

RIFDISCOVERER: A TOOL FOR FINDING RESOURCE INTERACTION FAILURES

Summary

- Introduction
- The RIFDiscoverer tool
  - Architecture
  - Design and Implementation
  - Test Instrumentation
- Preliminary Results

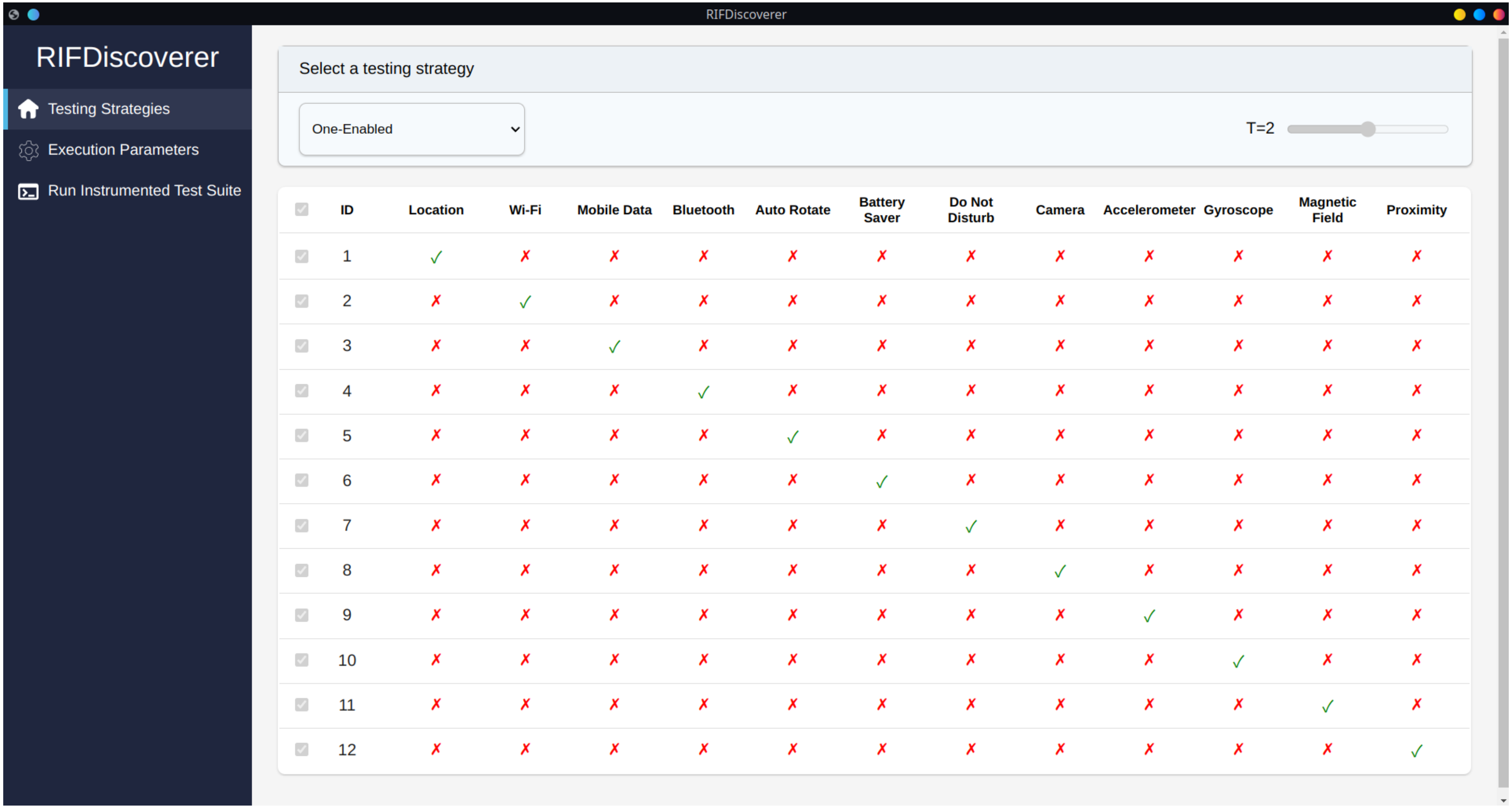
Introduction

- Mobile devices have a rich set of resources
- “Resource” refers to sensors, radios, and user-controlled options
- User interaction with devices can enable or disable the resources
- Unexpected application behavior can occur in specific resource settings
- However, the testing of all input combinations is impracticable

Resource Interactions

- Resource interactions are like feature interactions
- Sampling strategies are used for decreasing the testing effort
- RIFDiscoverer
  - Tool for helping developers and testers to deal with Resource Interaction Failures

The RIFDiscoverer Tool



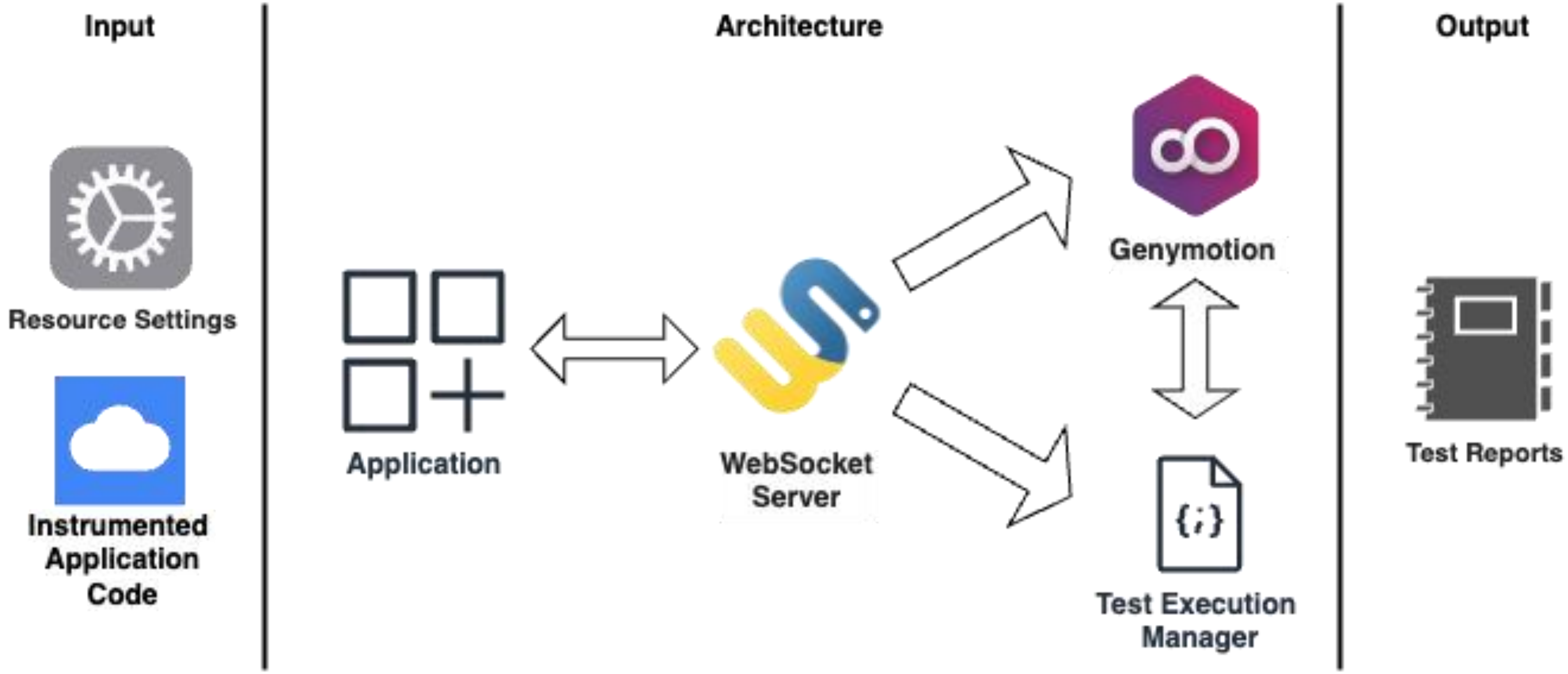
Select a testing strategy

One-Enabled

T=2

☑	ID	Location	Wi-Fi	Mobile Data	Bluetooth	Auto Rotate	Battery Saver	Do Not Disturb	Camera	Accelerometer	Gyroscope	Magnetic Field	Proximity
☑	1	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
☑	2	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
☑	3	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
☑	4	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
☑	5	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
☑	6	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
☑	7	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
☑	8	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
☑	9	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
☑	10	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
☑	11	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
☑	12	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Archicteture



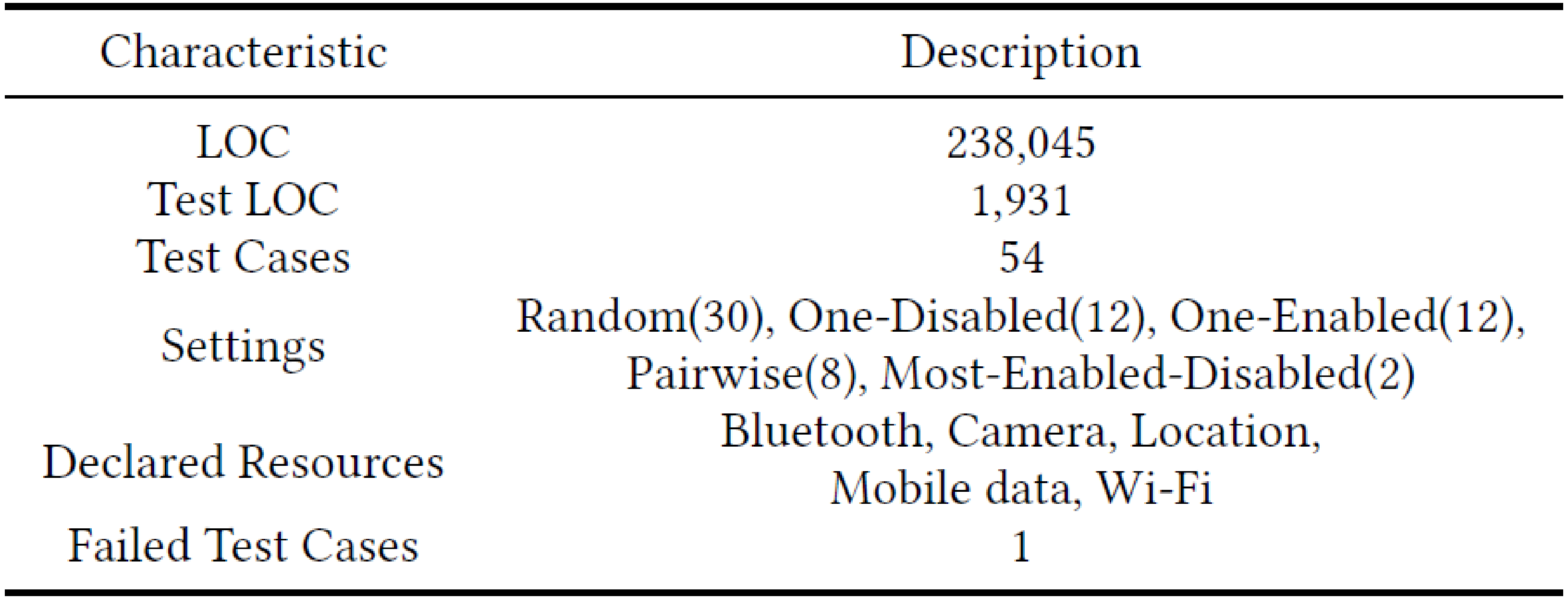
Design and Implementation

- Front-end implemented in Python EEL
  - Direct call to Python code
- WebSocket server
  - Run in a separate thread
  - Tool responsiveness

Test Instrumentation

- Functional tests are the target
  - Android APIs for interacting with the device
- Extension by means of UI Automator
- Each test class is extended with instrumentation code
- Before each test case the instrumentation code is executed
- Test reports are processed

Preliminary Results – Threema app



Characteristic	Description
LOC	238,045
Test LOC	1,931
Test Cases	54
Settings	Random(30), One-Disabled(12), One-Enabled(12), Pairwise(8), Most-Enabled-Disabled(2)
Declared Resources	Bluetooth, Camera, Location, Mobile data, Wi-Fi
Failed Test Cases	1

Questions?