UFMG

lab▪soft
software engineering laboratory

Applying Spectrum-Based Fault Localization to Android Applications

Euler Horta Marinho

Fischer Ferreira

João P. Diniz

Eduardo Figueiredo

(XXXVII Brazilian Symposium on Software Engineering)

Software Engineering Lab (LabSoft)
https://labsoft-ufmg.github.io

Summary

- Introduction
- Background
- Study Design
- Results
- Conclusion

Introduction

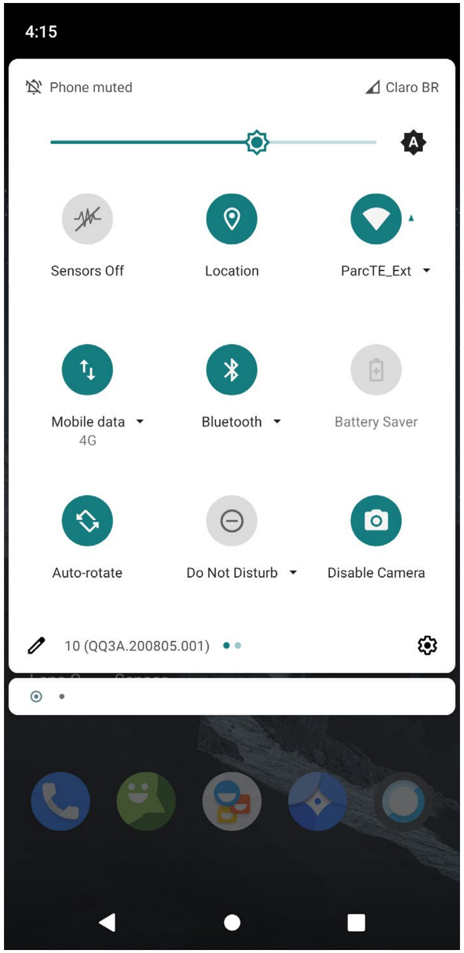
- Testing one of the most used QA approach

- Debugging is another QA approach
  - Aiming to the localization and removal of faults
  - Manual debugging can be extremely challenging

- Fault localization techniques
  - Spectrum-Based Fault Localization (SBFL)

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



Goal

- Evaluate the use of SBFL in Android applications
  - Use faults seeded from mutation operators
  - Ochiai coefficient as an indicator of suspicious faulty code (Abreu et al. 2016)

- Verify the sensitivity of SBFL to resource interaction failures
  - Failures of the study of Marinho et al. 2023

Background

SBFL techniques

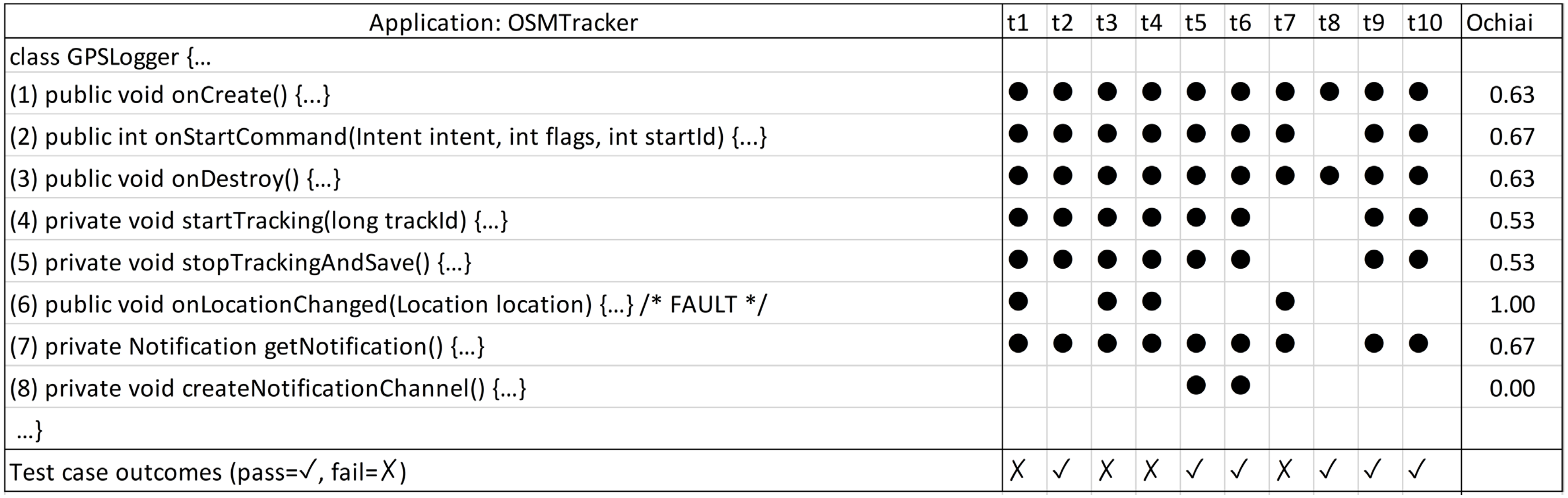
- Analysis of the program spectra (test coverage)
  - Statements, blocks, predicates, **methods**

- Produces a ranked list of elements in descending order of suspiciousness

- Ochiai is considered one of the best performance metrics

- Intuitively, the more a program element is executed by failing tests the more suspicious it is

Example of Ochiai coefficient



Application: OSMTracker	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	Ochiai
class GPSLogger {...											
(1) public void onCreate() {...}	●	●	●	●	●	●	●	●	●	●	0.63
(2) public int onStartCommand(Intent intent, int flags, int startId) {...}	●	●	●	●	●	●	●		●	●	0.67
(3) public void onDestroy() {...}	●	●	●	●	●	●	●	●	●	●	0.63
(4) private void startTracking(long trackId) {...}	●	●	●	●	●	●			●	●	0.53
(5) private void stopTrackingAndSave() {...}	●	●	●	●	●	●			●	●	0.53
(6) public void onLocationChanged(Location location) {...} /* FAULT */	●		●	●			●				1.00
(7) private Notification getNotification() {...}	●	●	●	●	●	●	●		●	●	0.67
(8) private void createNotificationChannel() {...}					●	●					0.00
...}											
Test case outcomes (pass=✓, fail=✗)	✗	✓	✗	✗	✓	✓	✗	✓	✓	✓	

Resource interaction failures

- Applications with unexpected behaviors
  - Manifested in certain combinations of enabled/disabled resources
- Settings are tuples of pairs <resource, state>

Auto Rotate, !Wi-Fi, Battery_Saver, Accelerometer, Bluetooth, Gyroscope, Camera, Light, Do_Not_Disturb, Magnetometer, !Location, Orientation, Mobile_Data, Proximity

Previous studies on this subject

- High number of input settings

- Marinho et al. (2021)
  - 8 resources (256 settings); 10 applications

- Marinho et al. (2023)
  - Sampling testing strategies (Random, One Enabled, One Disabled, Most Enabled Disabled, Pairwise)
  - 14 resources (> 16K settings); 20 applications

Study Design

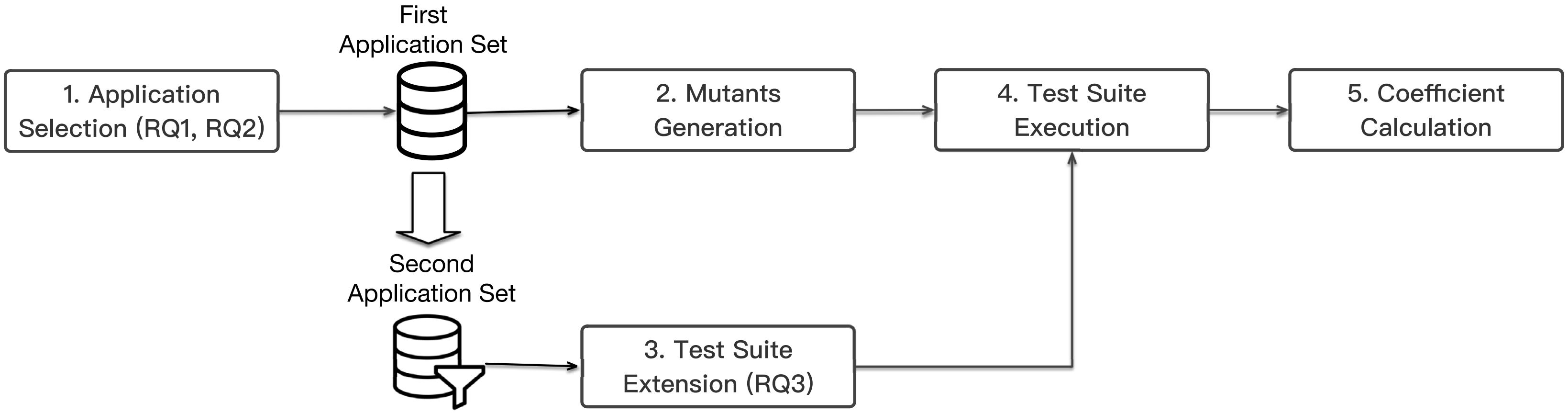
Research Questions

- □ RQ1: To what extent SBFL can be used for mobile applications?

- □ RQ2: How different is the ranking coefficient for faults in resource related classes and faults in general classes?

- □ RQ3: How sensitive is SBFL to variations in resource settings?

Steps of the study



1. Application Selection (RQ1, RQ2)

First Application Set

2. Mutants Generation

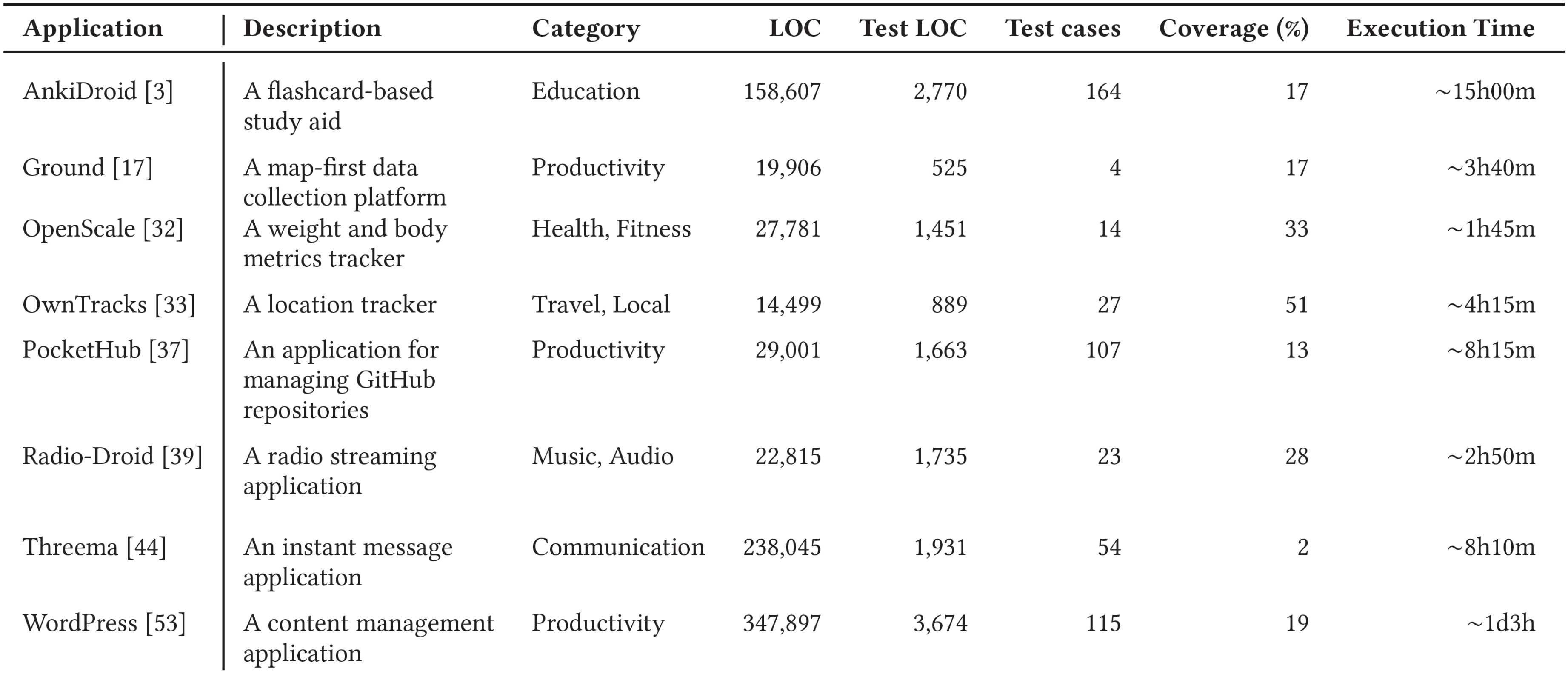
4. Test Suite Execution

5. Coefficient Calculation

Second Application Set

3. Test Suite Extension (RQ3)

1. Application selection



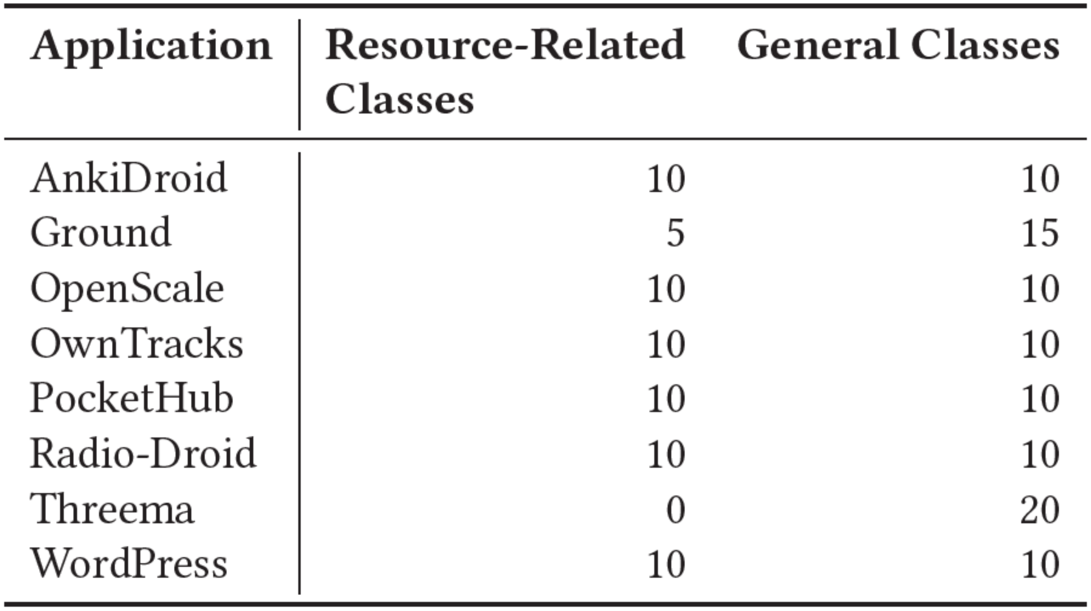
Application	Description	Category	LOC	Test LOC	Test cases	Coverage (%)	Execution Time
AnkiDroid [3]	A flashcard-based study aid	Education	158,607	2,770	164	17	~15h00m
Ground [17]	A map-first data collection platform	Productivity	19,906	525	4	17	~3h40m
OpenScale [32]	A weight and body metrics tracker	Health, Fitness	27,781	1,451	14	33	~1h45m
OwnTracks [33]	A location tracker	Travel, Local	14,499	889	27	51	~4h15m
PocketHub [37]	An application for managing GitHub repositories	Productivity	29,001	1,663	107	13	~8h15m
Radio-Droid [39]	A radio streaming application	Music, Audio	22,815	1,735	23	28	~2h50m
Threema [44]	An instant message application	Communication	238,045	1,931	54	2	~8h10m
WordPress [53]	A content management application	Productivity	347,897	3,674	115	19	~1d3h

508 **~71h**

2. Mutants generation

- Mutants generation using the tool presented in the study of Diniz et al. (2021)
  - Four mutant operators (AOR, ROR, LCR, SBR)
- Resource-related classes identified analyzing the imported packages
  - Study of Oliveira et al. (2022)

Generated mutants



Application	Resource-Related Classes	General Classes
AnkiDroid	10	10
Ground	5	15
OpenScale	10	10
OwnTracks	10	10
PocketHub	10	10
Radio-Droid	10	10
Threema	0	20
WordPress	10	10

3. Test suite extension

- Same strategy of Marinho et al. (2023)
  - OwnTracks, PocketHub, Threema
- Instrumented code aiming to control 14 common resources

Auto rotate	Wi-Fi
Battery saver	Accelerometer
Buetooth	Gyroscope
Camera	Light
Do not disturb	Magnetometer
Location	Orientation
Mobile data	Proximity

4. Test execution

- Test suites executed in a real device with code coverage enabled
  - Each test need to be executed separately

- Experimental effort ranging from 1h45m (OpenScale) to 1d3h (WordPress)

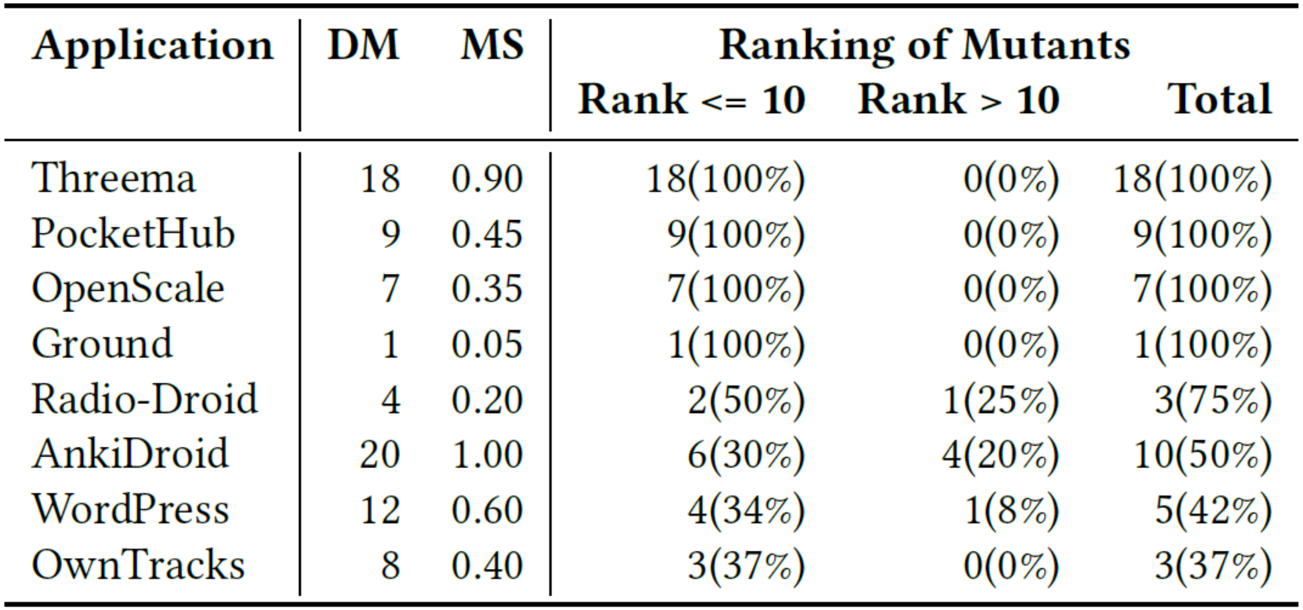
5. Coefficient calculation

- Test reports (test results and test coverage) were parsed to get needed information

- Ochiai calculated at the method-level

Results

RQ1 – Use of SBFL for mobile apps

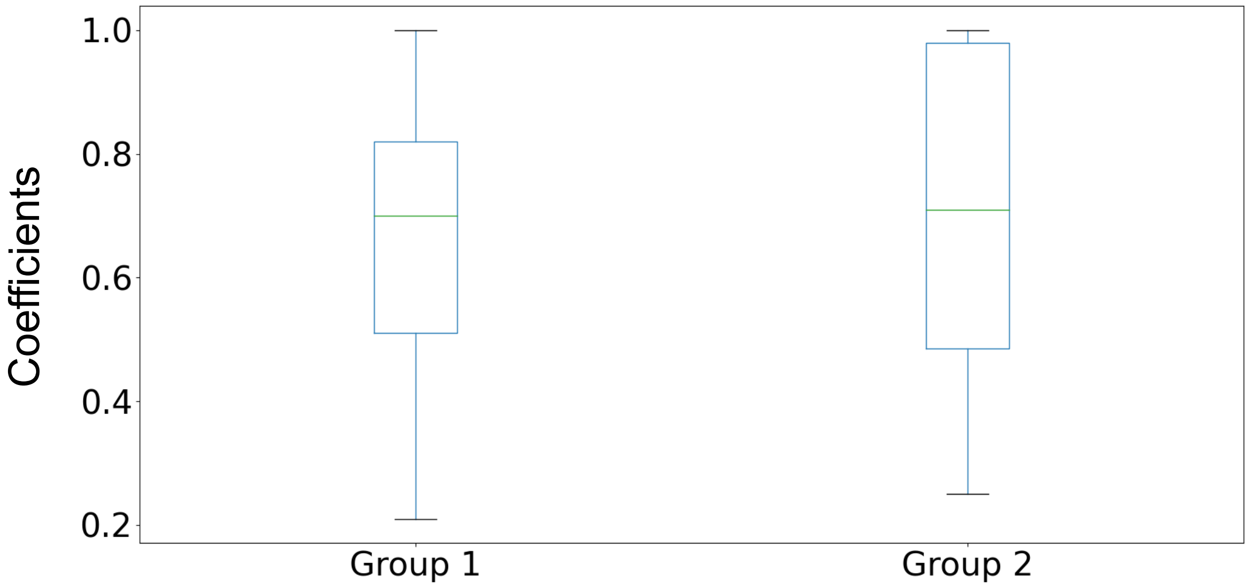


Application	DM	MS	Ranking of Mutants		
			Rank <= 10	Rank > 10	Total
Threema	18	0.90	18(100%)	0(0%)	18(100%)
PocketHub	9	0.45	9(100%)	0(0%)	9(100%)
OpenScale	7	0.35	7(100%)	0(0%)	7(100%)
Ground	1	0.05	1(100%)	0(0%)	1(100%)
Radio-Droid	4	0.20	2(50%)	1(25%)	3(75%)
AnkiDroid	20	1.00	6(30%)	4(20%)	10(50%)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

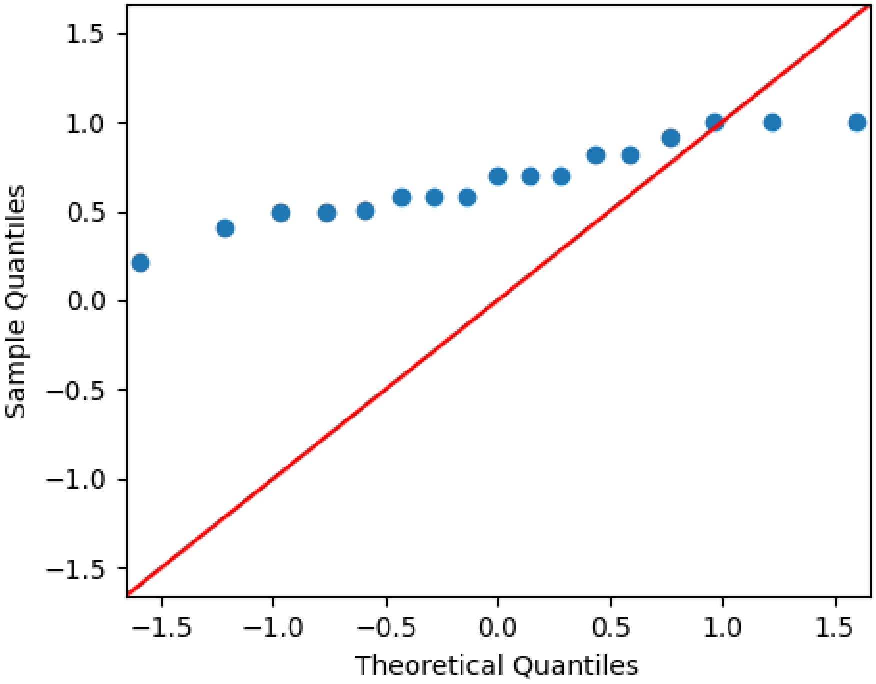
* DM = Dead mutants
* MS = Mutation score

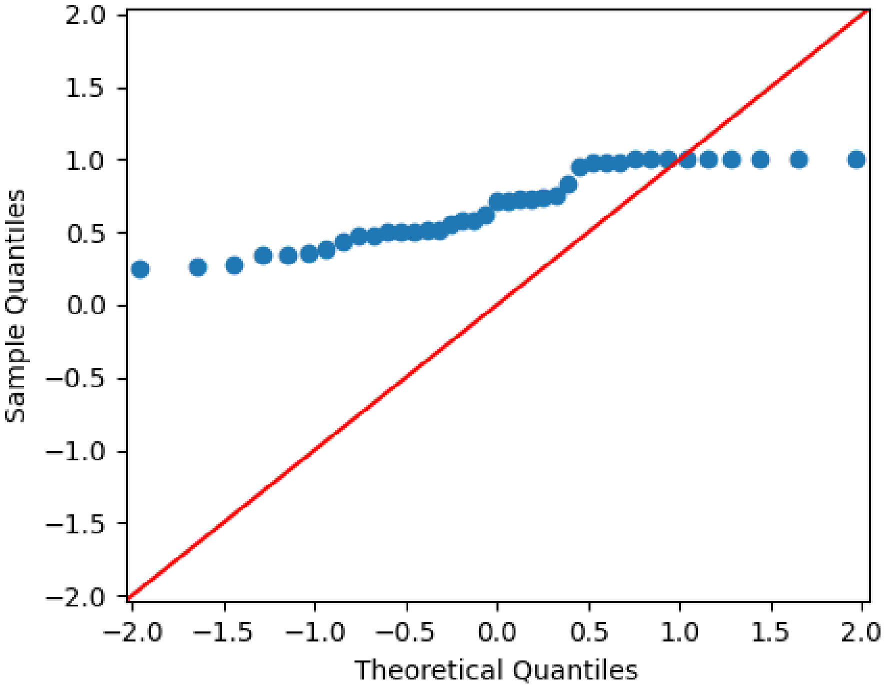
RQ2 – Ochiai for two groups of classes

- □ Coefficients of Group1 (Resource-related classes) and Group2 (General classes)



Normality test





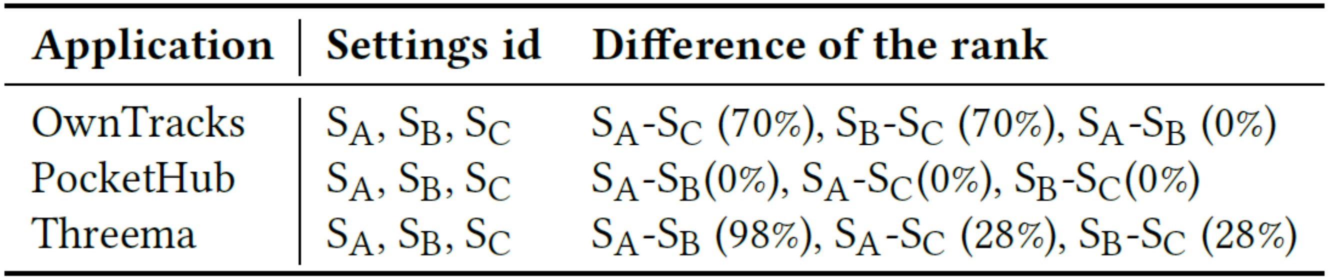
Nonparametric test

- Mann-Whitney U test
  - H0: Groups 1 and 2 are from the same population
  - H1: Groups 1 and 2 are not from the same population

- 5% confidence interval (p-value = 0.99)
  - Does not allow the rejection of the null hypothesis
  - There is no evidence of a difference between the groups

RQ3 – Sensitivity to variations in resources settings

- Three applications with failures in three executions
  - Settings associated to this kind of failure
  - Same failure set



Application	Settings id	Difference of the rank
OwnTracks	S_A, S_B, S_C	S_A-S_C (70%), S_B-S_C (70%), S_A-S_B (0%)
PocketHub	S_A, S_B, S_C	S_A-S_B(0%), S_A-S_C(0%), S_B-S_C(0%)
Threema	S_A, S_B, S_C	S_A-S_B (98%), S_A-S_C (28%), S_B-S_C (28%)

Conclusion

Conclusion

- SBFL was able to rank more than 75% of fault code in 5 out of 8 applications
- For the same failure (mutant), ranking depends on the combination of enabled resources
- Future studies

UFMG

lab·soft
software engineering laboratory

Questions?