4th International Workshop on Refactoring (IWoR 2020)

An Exploratory Study on the Refactoring of Unit Test Files in Android Applications

Anthony Peruma, Christian D. Newman, Mohamed Wiem Mkaouer, Ali Ouni, Fabio Palomba

RIT Rochester Institute of Technology

ÉTS École de technologie supérieure

Università degli Studi di Salerno

REFACTORING

An essential software maintenance activity

Refactoring operations enable developers to take necessary actions to correct bad programming practices

Smelly source code files

Smells are symptoms of bad programming practices – a threat to design

Refactoring source code

Refactoring improves the internal design of software systems without altering its external behavior

Smell free source code

Refactoring involves locating and correcting smells exhibited by the source code

TEST SMELLS

Test code, like production code, is subject to smells

Formally introduced in 2001 with 11 smell types

Inclusion of additional smell types, analysis of their evolution and longevity, and elimination patterns

Tools to detect specific smell types

Studies on traditional Java applications

2.5 million+ apps available on Google Play (December 2019)

GOAL

Expand our understanding of the **relationship between refactoring** changes and their **effect on test smells** in Android apps

RESEARCH QUESTIONS

01

What types of refactoring operations are applied to unit test files compared to non-test files?

02

What types of refactoring operations are frequently applied to smelly test files?

03

What kinds of refactorings are typically used to remove test smells?

CONTRIBUTIONS

01

An understanding of refactoring operations applied to test suites of Android apps

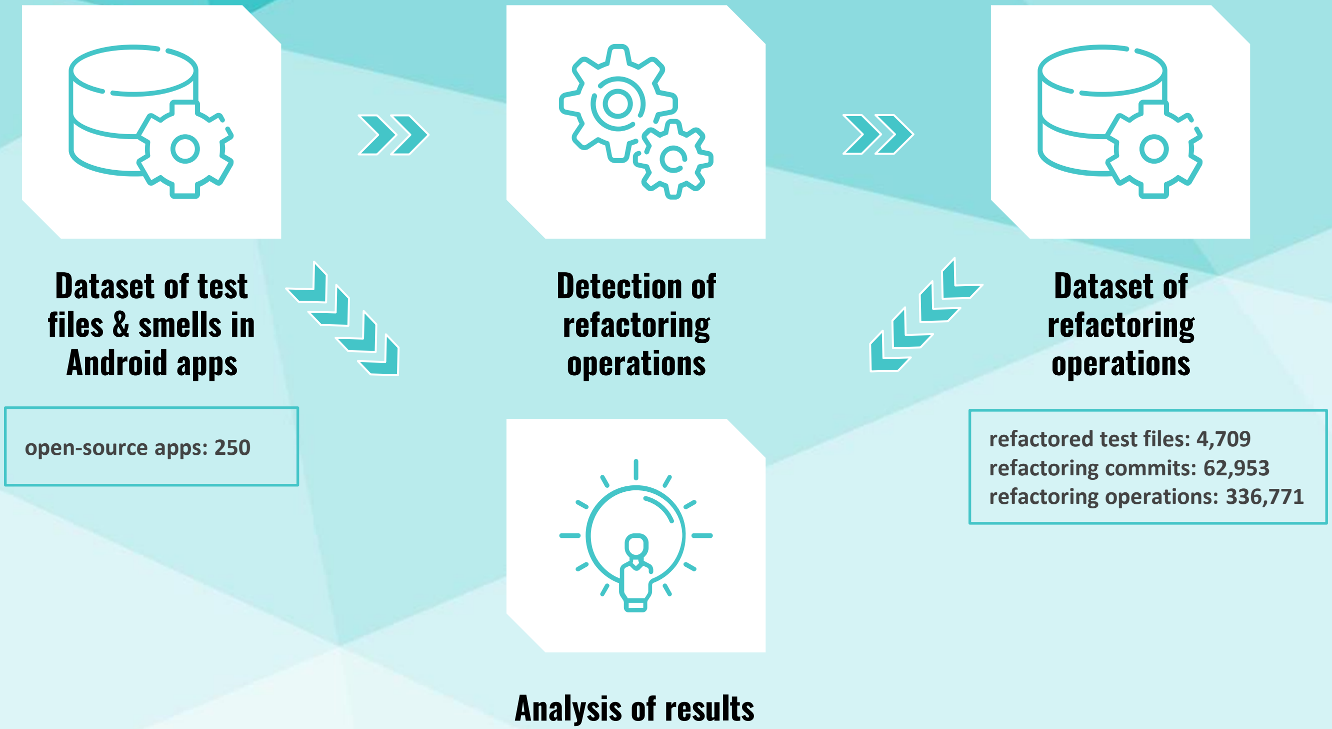
02

Insights into the relationships between refactoring operations and test smells

03

A dataset for replication and extension purposes, available on our project website

EXPERIMENT DESIGN



Dataset of test files & smells in Android apps

open-source apps: 250

Detection of refactoring operations

Dataset of refactoring operations

refactored test files: 4,709
refactoring commits: 62,953
refactoring operations: 336,771

Analysis of results

TEST SMELLS & REFACTORING OPERATIONS

19 Test Smells

- Assertion Roulette
- Conditional Test Logic
- Constructor Initialization
- Default Test
- Duplicate Assert
- Eager Test
- Empty Test
- Exception Handling
- General Fixture
- Ignored Test

.
.
.

39 Refactoring Operations

- Rename Method
- Change Variable Type
- Rename Variable
- Inline Method
- Extract Method
- Move Class
- Move Method
- Pull Up Method
- Split Attribute
- Push Down Method.

.
.
.

RQ $_1$ - What types of refactoring operations are applied to unit test files compared to non-test files?

Test Files

Refactoring Operations	Count	%
Rename Method	1511	19.81%
Change Variable Type	1452	19.03%
Rename Variable	803	10.53%
Change Attribute Type	773	10.13%
Extract Method	426	5.58%
Other Operations	2,664	34.29%
Total	7,629	100%

65%

Non-Test Files

Refactoring Operations	Count	%
Move Class	23,180	13.33%
Change Parameter Type	14,178	8.15%
Change Attribute Type	12,921	7.43%
Rename Method	12,074	6.94%
Rename Parameter	11,299	6.50%
Other Operations	100,249	57.65%
Total	173,901	100%

42%

- 91.98% of refactorings are applied to methods Vs. 8.02% to classes
- Test files tend to undergo more renames

- 49.27% of refactorings are applied to methods Vs. 50.73% to classes
- Non-test files are subject to more design level types of refactorings

RQ $_2$ - What types of refactoring operations are frequently applied to smelly test files?

Approach: Extracted test files that exhibited only one smell type and then looked at the refactorings in the file

4,589 test files that had one or more smells had undergone a refactoring

Co-occurring			
Smell Type	Refactoring Operation	Count	%
Assertion Roulette	Change Variable Type	141	53.01%
Eager Test	Extract Method	14	42.42%
Lazy Test	Extract Method	20	30.30%
General Fixture	Change Attribute Type	8	38.10%
Redundant Assertion	Move Method	9	29.03%

The variable and the smell exist within a test method

These smells and refactoring operation are related to test methods

The attribute is utilized in the setup() method

A debugging smell introduced when making design level changes

RQ $_3$ - What kinds of refactorings are typically used to remove test smells?

Approach: Lifetime history of a smelly test file; matched the refactoring applied with a smell reduction

481

Smelly test files that had a reduction & refactoring

1.30

Avg. smells removed by refactoring test file

2.12

Avg. refactoring operations to reduce smell count

38%

Freq. for a single refactoring to remove a single smell type

- Change Variable Type being one of the most common refactorings applied when a smell is removed
- Eager Test is frequently resolved by developers when performing a single refactoring operation
- Please refer the paper for a qualitative based set of examples on the co-occurrence of smell type reduction and refactoring operations -- Extract Method resolving the Conditional Test Logic smell
- Manual review - most refactorings are applied for reasons other than for the correction of smell
  - More in-depth, and developer supported, studies are needed

TAKEAWAYS

RQ 1

Developers are better prepared to estimate rework effort for (non-) test files

Specialized tools for refactoring files

RQ 2

Developers are better prepared in determining the most likely smell being introduced when performing a refactoring of a test file

RQ 3

Refactoring of test files are mostly related to development activities such as fixing issues, adhering to coding standards, etc.

SUMMARY

01 Mined refactoring operations and test smells in 250 open-source Android apps

02 App developers apply a different set of refactorings to test and non-test source code files

03 Certain test smells and refactoring operations that co-occur frequently (e.g., Lazy Test & Extract Method)

04 There exist scenarios where refactoring operations are utilized to correct a test smell

THANKS!

https://testsmells.github.io