Towards automated refactoring of smelly test code

Railana Santana Lago

Advisor: Ivan Machado

WEPGCOMP 2023

Ficha do trabalho

Título	Towards automated refactoring of smelly test code
Nome	Railana Santana Lago
Curso	Doutorado
Orientador(es)	Ivan Machado
Ingresso	OUT/2020
Qualificação	2024.1
Defesa	A definir
Bolsista?	Sim CAPES

Context

- Software testing
  - Improve the quality of *production code*
- Test code is considered **second-class** code
- Production code quality **vs.** test code quality
- Test code is susceptible to design antipatterns
  - ***Test smells*** [1]**:** symptoms and indications of design problems in the test code that impair understanding, readability and consequently the maintenance of the test.
- Refactoring test smells has **not** been a *priority* for engineers [2] and **is not trivial** [3]

[1] Van Deursen, Arie, et al. "**Refactoring test code.**" Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP2001), 2001.
[2] Junior, N. S., Rocha, L., Martins, L. A., & Machado, I. (2020). **A survey on test practitioners' awareness of test smells.**
[3] SHRIVASTAVA, D. P., & JAIN, R. **Improve The Test Case Design of Object Oriented Software by Refactoring**. 2010. International Journal of Computer Science and Information Security (IJCSIS)

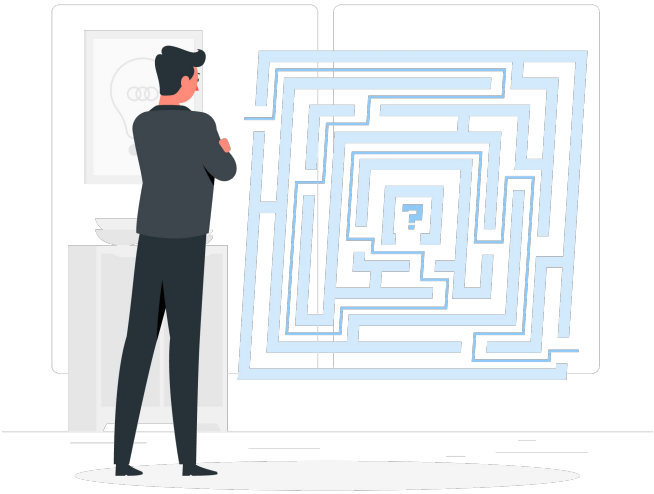
Context

- *Variety* of types of test smells[1]
- Each **test smell** has its particularities and each type affects the test code in a different way
  - ***Sleep test*** [2]**:** May slow down test execution.
  - ***Conditional Test Logic*** [2]**:** Increases test complexity.
- Several approaches to fix test smells have been proposed in the literature
  - Existing approaches seek to *progressively* fix test smells
  - Refactoring *individual* test smell instances
  - One test smells at a time, until multiple changes are applied to the test code and the code is completely refactored.

[1] Garousi, V., & Küçük, B. (2018). **Smells in software test code: A survey of knowledge in industry and academia.** Journal of systems and software, 138, 52-81.
[3] Peruma, A., Almalki, K., Newman, C. D., Mkaouer, M. W., Ouni, A., & Palomba, F. (2020, November). **Tsdetect: An open source test smells detection tool.** In Proceedings of the 28th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering (pp. 1650-1654).

Problem and Motivation

- A **single** smell instance **x** **Multiple** test smell instances [1]
- Multiple test smell instances (test smells **co-occurrence**) [2]
  - Some test smells types tend to co-exist with others
- Frequent co-occurrence of different design problems in tests[3]
  - All test smells co-occur with the *Assertion Roulette* test smell;
  - *Mystery Guest* and *Resource Optimism;*
  - *Mystery Guest* and *Indirect pairs Testing;*
  - *Indirect Testing* and *Test Code Duplication test smells*.



[1] M. Abbes, F. Khomh, Y.-G. Guéhéneuc, and G. Antoniol, "**An empirical study of the impact of two antipatterns, blob and spaghetti code, on program comprehension,**" in 2011 15th European Conference on Software Maintenance and Reengineering, 2011, pp. 181–190.

[2] D. Spadini, F. Palomba, A. Zaidman, M. Bruntink, and A. Bacchelli, "**On the relation of test smells to software code quality,**" in 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2018, pp. 1–12.

[3] F. Palomba, D. Di Nucci, A. Panichella, R. Oliveto, and A. De Lucia, "**On the diffusion of test smells in automatically generated test code: An empirical study,**" in Proceedings of the 9th International Workshop on Search-Based Software Testing, ser. SBST '16. ACM, 2016, p. 5–14.

Problem and Motivation

Problem: Current research *only* investigates and proposes approaches to refactoring *individual instances* of test smells.

- As test smells rarely appear in isolation in tests, it is necessary to investigate different strategies to refactor test code with **multiple test smells** quickly and safely.
- Refactoring guidelines capable of **removing** more than one test smell instance can reduce the amount of code changes and speed up the refactoring process.
- Trivial examples:
  - *Assertion Roulette* and *Duplicate Assert* - Add Assertion Explanation or extract asserts.
  - *Mystery Guest* and *Resource Optimism* - Inline Resource refactoring, removing external dependency.
  - *Empty Test* is also an *Unknown Test* - Add a test.

Objective

The objective of this study is to **investigate** how multiple instances of test smells usually interact to each other in open source projects and how these *multiple instances* can be refactored.

- **OE1)** Survey and synthesize evidence available in the literature on test code refactoring.
- **OE2)** Identify and empirically evaluate how *co-occurrences* of test smells affect the quality of test code.
  - **OE2.1)** Characterize the *effects* of the occurrence of test smells on the quality of the test code.
- **OE3)** Find test *refactorings* that occur simultaneously.
- **OE4)** Propose and evaluate an *approach to refactoring* tests with multiple test smells.

Search strategy

Strategy

Systematic literature review

Exploratory longitudinal study with open source projects

Build guidelines for refactoring multiple test smells

Preparation of the refactoring catalog

Implementation of tool with guidelines

Empirical evaluation of tool with practitioners

Objective

OE1) Survey and synthesize evidence available in the literature on test code refactoring.

OE2) Identify and empirically evaluate how *co-occurrences* of test smells affect the quality of test code.

OE2.1) Characterize the *effects* of the occurrence of test smells on the quality of the test code.

OE3) Find test *refactorings* that occur simultaneously.

OE4) Propose and evaluate an *approach to refactoring* tests with multiple test smells.

Systematic literature review

Research Questions (RQ):

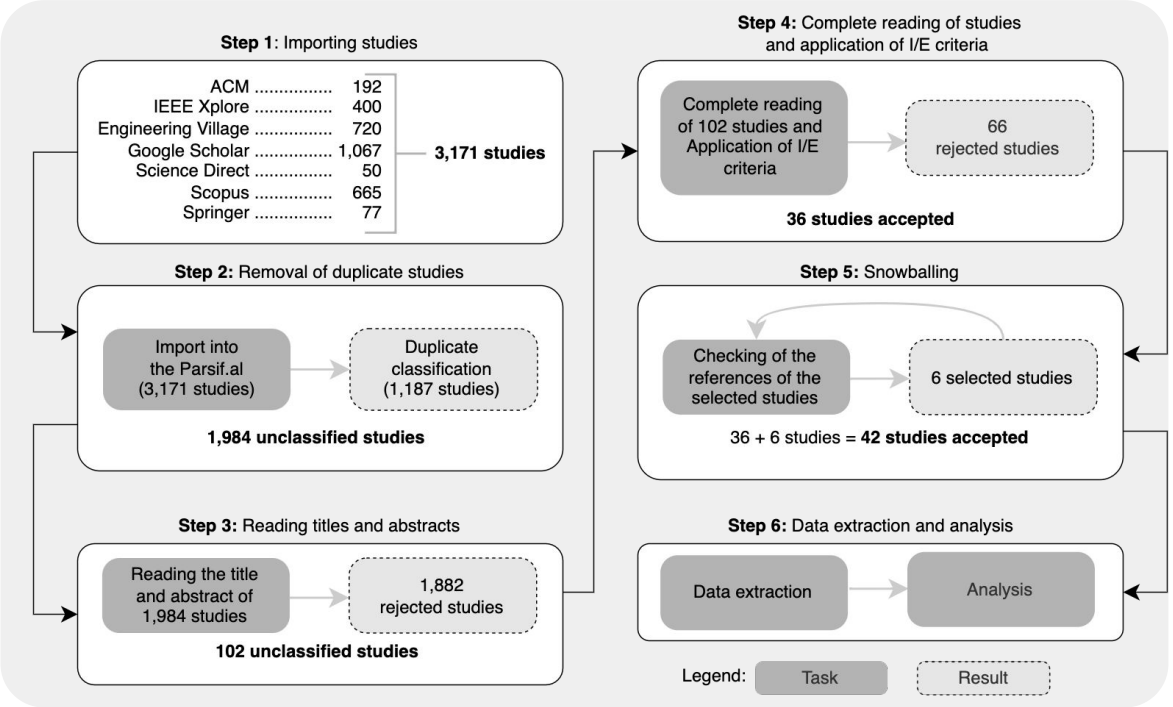
RQ1) What are the existing activities for test code refactoring?

RQ2) What types of test code issues can be resolved using refactoring?

RQ3) What automated tools are available for test code refactoring?

RQ4) What empirical methods were employed in primary studies?

Steps



Search string

software AND (test OR testing) AND (refactoring OR refactor)

Systematic literature review

Results

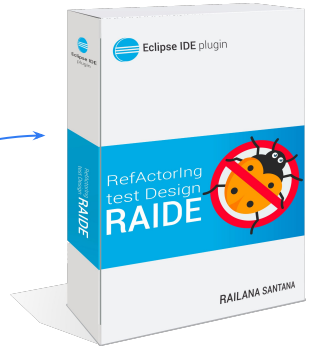
- We identified **190** test code problems, **211** refactorings, and **26** tools
- The majority of the studies presented generic refactorings suitable for both production and test code
- **Few** studies focused exclusively on test code design
- Some anti-patterns had **many** refactoring strategies, while others **lacked a clear** strategy
- Existing tools have not evolved significantly since their inception, and many refactoring strategies *have not been evaluated*.
- Our study highlights the need for **more studies** on *recommending refactorings* for test code and investigating the effectiveness and effects of each refactoring approach.

Exploratory study - Pilot

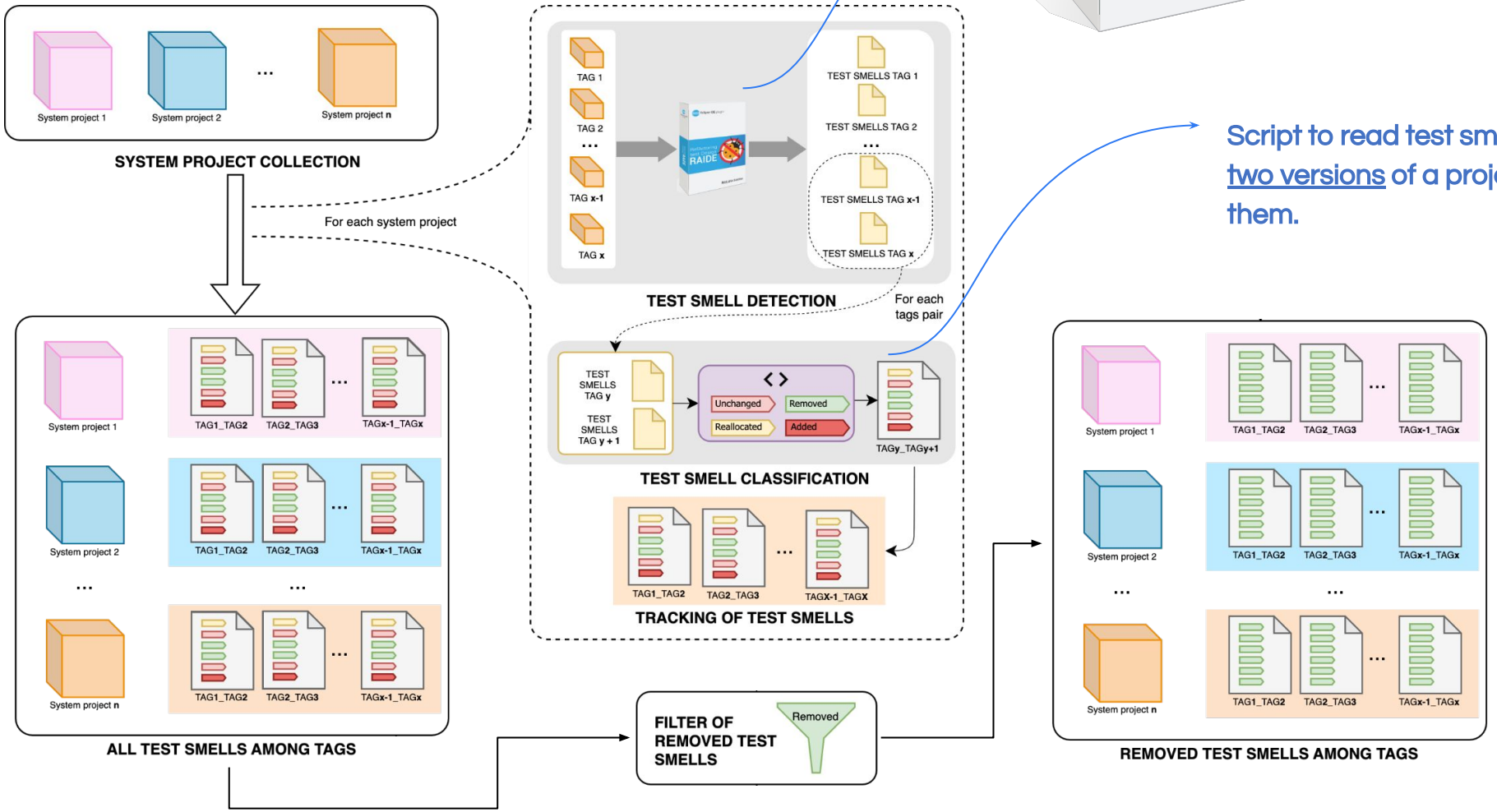
- Apache Maven Dependency Plugin (AMDP)[1] project
- We detected test smells in AMDP project with RAIDE (20 test smells types)
- We selected two versions of AMDP project
- We **manually** classified the test smells in two versions:
  - Added, Unchanged, Reallocated, and Removed
- We identified (and them analyzed) **21 test smells** (removed)
- A *wrong* refactoring attempt may **unintentionally** move a test smell or insert new ones
- Three methods yield the largest amount of **simultaneous test smells changes**
- If test smells can be *reallocated together*, then it is possible that they also can be refactored through simultaneous changes.

[1] https://maven.apache.org/plugins/maven-dependency-plugin/

Exploratory longitudinal study



RAIDE detects new test smells



Script to read test smell reports from <u>two versions</u> of a project and classify them.

Status

Article submitted to the **ACM CSUR** journal.

Systematic literature review

Exploratory longitudinal study with open source projects

Build guidelines for refactoring multiple test smells

Preparation of the refactoring catalog

Implementation of tool with guidelines

Empirical evaluation of tool with practitioners

OE1) Survey and synthesize evidence available in the literature on test code refactoring.

OE2) Identify and empirically evaluate how *co-occurrences* of test smells affect the quality of test code.

OE2.1) Characterize the *effects* of the occurrence of test smells on the quality of the test code.

OE3) Find test *refactorings* that occur simultaneously.

OE4) Propose and evaluate an *approach to refactoring* tests with multiple test smells.

Status

1. Pilot Study
2. RAIDE
3. SCRIPT
4. Design and planning
5. Execution
6. Analysis

Build guidelines for refactoring multiple test smells

Preparation of the refactoring catalog

Implementation of tool with guidelines

Empirical evaluation of tool with practitioners

Systematic literature review

Exploratory longitudinal study with open source projects

OE1) Survey and synthesize evidence available in the literature on test code refactoring.

OE2) Identify and empirically evaluate how *co-occurrences* of test smells affect the quality of test code.

OE3) Find test *refactorings* that occur simultaneously.

OE4) Propose and evaluate an *approach to refactoring* tests with multiple test smells.

OE2.1) Characterize the *effects* of the occurrence of test smells on the quality of the test code.

Status

Systematic literature review

Exploratory longitudinal study with open source projects

OE1) Survey and synthesize evidence available in the literature on test code refactoring.

OE2) Identify and empirically evaluate how *co-occurrences* of test smells affect the quality of test code.

OE2.1) Characterize the *effects* of the occurrence of test smells on the quality of the test code.

OE3) Find test *refactorings* that occur simultaneously.

OE4) Propose and evaluate an *approach to refactoring* tests with multiple test smells.

Build guidelines for refactoring multiple test smells	Pending
Preparation of the refactoring catalog	Pending
Implementation of tool with guidelines	In progress
Empirical evaluation of tool with practitioners	Pending

Thank you!

Questions?

railana.santana@ufba.br