Ensino da Prática de Refatoração de *Test Smells*

Humberto Damasceno, Carla Bezerra

[1]Universidade Federal do Ceará (UFC) – Campus Quixadá

hdamasceno1998@gmail.com, carlailane@ufc.br

Abstract. *Test smells represent a set of poorly designed tests, which can harm test code maintenance. Although fundamental steps have been taken to understand test smells, there is still an evident lack of studies on teaching the practice of refactoring test smells. This paper presents an experience report in teaching test smells refactoring with 30 students from the Software Engineering course's Software Quality and Software Maintenance classes. We investigated: (i) the main skills acquired by students during the refactoring of test smells and (ii) the difficulty of refactoring these test smells by students. Our findings highlight the benefits of teaching test smells refactoring and a methodology for teaching these concepts in practice.*

Resumo. *Test smells representam um conjunto de testes mal projetados, que podem prejudicar a manutenção do código de teste. Embora etapas fundamentais para entender os test smells tenham sido tomadas, ainda há uma evidente falta de estudos no ensino da prática de refatoração de test smells. Este artigo apresenta o relato de experiência no ensino da refatoração de test smells com 30 estudantes das turmas de Qualidade e Manutenção de software do curso de Engenharia de Software. Nós investigamos: (i) as principais habilidades adquiridas pelos alunos durante a refatoração dos test smells e (ii) a dificuldade de refatoração desses test smells pelos alunos. Nossas descobertas apontam os benefícios do ensino da refatoração de test smells e uma metodologia para ensino desses conceitos na prática.*

1. Introdução

Teste de software é uma atividade crucial na garantia da qualidade de um sistema. No entanto, essa atividade pode exigir muito esforço e recursos caros, especialmente quando executado manualmente [Orso e Rothermel 2014]. Neste contexto, os testes automatizados implementados através de *frameworks* ou ferramentas de teste, tornaram-se alternativas de primeira classe desempenhando um papel particularmente importante no processo de garantia de qualidade de software [Myers et al. 2011]. Testes automatizados podem auxiliar na identificação e remoção de erros e garantir que o código de produção seja robusto em diversas condições de uso [Candea et al. 2010].

No entanto, seja em testes desenvolvidos manualmente ou automatizados, os desenvolvedores de software devem ter cuidado com o código de teste, assim como com o código de produção [Berner et al. 2005, Beller et al. 2019]. Estudos recentes mostram que os desenvolvedores entendem e dão mais importância ao código de produção do que ao código de teste, causando assim problemas na qualidade do teste [Beller et al. 2015, Palomba et al. 2018]. De fato, testes mal projetados podem resultar em falta de capacidade de manutenção e possíveis falhas no software testado [Yusifoğlu et al. 2015].

[Van Deursen et al. 2001] introduziram o conceito de *test smells*, que representam um teste mal projetado que pode prejudicar a legibilidade, manutenção e qualidade do código de teste. Desde sua conceituação, os *test smells* tornaram-se foco de diversos estudos na área de teste de software devido à sua importância para a qualidade do código de teste [Bavota et al. 2012, Kim 2020]. Alguns estudos indicam que os *test smells* afetam negativamente a manutenção e a compreensão do código de teste [Bavota et al. 2012, Spadini et al. 2018]. Portanto, é necessário entender seus prováveis efeitos e propor mecanismos, estratégias e ferramentas para mitigá-los. A refatoração de software consiste em pequenas transformações de código para melhorar a qualidade do código-fonte sem comprometer sua funcionalidade geral e comportamento observável [Alizadeh et al. 2020, Fowler 1999, Paixão et al. 2020]. Os *test smells* podem ser removidos por meio de refatoração, o que pode impactar positivamente na qualidade do código de teste [Spadini et al. 2018, Campos et al. 2021].

Apesar da importância da detecção, refatoração e remoção dos *test smells* para qualidade do código de testes, pouco se tem dado atenção a esses conceitos no ensino de testes e na prática de testes na indústria [Bai et al. 2021]. Poucos trabalhos da literatura tem apresentado conteúdos e práticas para o ensino de refatoração de *test smells* [Aniche et al. 2019a, Bai et al. 2021, Aljedaani et al. 2023]. Isso também pode ser percebido nos conceitos apresentados no ensino de testes de software no currículo de cursos de computação [Garousi et al. 2020].

Dessa forma, o objetivo desse trabalho é apresentar um relato de experiência no ensino dos conceitos de detecção, refatoração e remoção dos *test smells*, analisando a qualidade do código de testes. A metodologia de ensino foi aplicada nas disciplinas de Qualidade de Software e Manutenção de Software do curso de Engenharia de Software da Universidade Federal do Ceará (UFC) - Campus Quixadá com 30 alunos no semestre de 2022.2. Para este artigo, nós analisamos: (i) as principais habilidades adquiridas pelos estudantes durante a refatoração dos *test smells*; e (ii) a dificuldade de refatoração dos *test smells* pelos alunos. Os resultados obtidos foram:

- O trabalho prático forneceu novas habilidades para os alunos, especialmente para resolução de problemas, pensamento crítico e habilidades analíticas.
- A refatoração dos *test smells* também se mostrou benéfica para o aprimoramento das habilidades interpessoais dos alunos.
- Os *test smells* que necessitam das técnicas de refatoração *Extract Method* ou *Inline Resource* para serem removidos, são os mais difíceis de se refatorar.
- Os *test smells Assertion Roulette* e *Magic Number Test* são mais fáceis de refatorar por necessitar de pouca alteração de código.

2. Fundamentação Teórica

2.1. *Test smells*

Presume-se que os *code smells* indicam um *design* ruim que leva a um código menos sustentável [Sjøberg et al. 2013]. Os *smells* podem ser encontrados tanto no código-fonte do projeto de software quanto no código de teste [Van Deursen et al. 2001]. Neste último, eles são frequentemente referidos como ***test smells***. *Test smells* referem-se a certos padrões em testes de software que podem indicar possíveis problemas ou pontos fracos nos testes. Eles são o efeito de testes mal projetados e podem resultar em

falta de capacidade de manutenção e possíveis falhas no software que está sendo testado [Yusifoğlu et al. 2015].

A literatura introduziu vários tipos de *test smells*. [Van Deursen et al. 2001] documentaram um conjunto inicial de onze *test smells*: *Assertion Roulette*, *Eager Test*, *For Testers Only*, *General Fixture*, *Indirect Testing*, *Lazy Test*, *Mystery Guest*, *Resource Optimism*, *Sensitive Equality*, *Test Code Duplication* e *Test Run War*. [Peruma et al. 2019] posteriormente, catalogou 12 novos tipos: *Conditional Test Logic*, *Constructor Initialisation*, *Default Test*, *Duplicate Assert*, *Empty Test* , *Exception Handling*, *Ignored Test*, *Magic Number Test*, *Redundant Assertion*, *Redundant Print*, *Sleepy Test, e Unknown Test*. Outros pesquisadores introduziram outro conjunto de *test smells*, principalmente com base em observações práticas, como [Garousi e Küçük 2018], que propuseram um catálogo diversificado de *test smells*, como resultado de uma revisão multivocal da literatura sobre *smells* no código de teste de software.

3. Trabalhos Relacionados

[Bai et al. 2021] realizaram um estudo exploratório para identificar as percepções dos alunos sobre testes unitários e os desafios encontrados durante a prática de codificação de tais testes. Os autores selecionaram 54 estudantes de duas universidades e forneceram a eles duas atividades. A primeira envolveu o *design* do teste de caixa preta, e a segunda englobou a implementação do teste de caixa branca. Os resultados obtidos apontam que os alunos acreditam que a cobertura de código é o resultado mais importante para suítes de teste. Em relação a prática dos testes, a maioria dos alunos foi incapaz de encontrar defeitos conhecidos. Através deste estudo, os autores identificaram conceitos de teste que requerem ênfase para um futuro mais eficaz em relação a estudos sobre testar o comportamento entre alunos. A diferença em relação a nosso estudo é que apresentamos um relato da experiência do ensino da refatoração de *test smells* e investigamos as habilidades adquiridas pelos alunos após a refatoração dos *smells*.

[Aljedaani et al. 2023] realizaram um experimento controlado com 96 graduandos em Ciência da Computação, com o intuito de investigar o impacto de dois dos mais comuns e difundidos tipos de *test smells* (*Assertion Roulette* e *Eager Test*) na capacidade dos alunos em depurar e solucionar problemas em casos de teste. As descobertas dos autores apontam que os alunos demoram mais tempo para corrigir erros no código de produção quando *test smells* ocorrem em seus respectivos casos de teste. Tal demora foi decorrente, na maioria dos casos, devido à ocorrência do *test smell Assertion Roulette*. Em nosso estudo, além dos *smells Assertion Roulette* e *Eager Test*, abordamos outros sete tipos de *test smells* em um experimento do ensino de refatoração com alunos de Engenharia de Software.

[Aniche et al. 2019b] descreveram o modo como adicionaram uma nova perspectiva pragmática ao curso de teste de software e exploraram os erros frequentes cometidos pelos alunos, tópicos difíceis de aprender, atividades de aprendizado favoritas e desafios enfrentados. Para tal, os autores analisamos os relatórios de *feedback* que a equipe de assistentes de ensino forneceu aos 230 alunos do curso de teste de software 2016-2017 na *Delft University of Technology*. Além disso, também pesquisaram 84 alunos e sete assistentes de ensino sobre suas respectivas percepções. Os resultados obtidos auxiliam os educadores não apenas a propor cursos pragmáticos de teste de software em disciplinas

de faculdade, mas também a comprender os desafios enfrentados pelos alunos durante os cursos de teste de software. Em nosso estudo, também avaliamos se as refatorações realizadas pelos estudantes foram assertivas, através da *JNose Test*, e a partir disso, conseguimos estabelecer estratégias específicas para o ensino de *test smells*.

[Buffardi e Aguirre-Ayala 2021] examinaram o trabalho dos alunos em tarefas de teste para investigar a inserção de *test smells* e práticas que indicam problemas potenciais problemas em testes unitários. Os autores descobriram três causas comuns de *test smells* nos testes unitários dos alunos: várias chamadas de função de membro, múltiplas *assertions* e *conditional logic*. Além disso, também exploraram como cada um poder estar associado a imprecisões no teste. Em um estudo semi-experimental, avaliaram a qualidade dos testes unitários dos alunos avaliando a precisão do teste e a confiabilidade dos testes em distinguir entre o corpo de um código de produção estável e um código contendo falhas. As análises correlacionais e comparativas revelaram que os testes de unidade com chamadas para funções de vários membros e/ou *conditional logic* foram associados a uma pior precisão do teste. A diferença em relação a nosso estudo, é que além de analisar as refatorações realizadas pelos alunos, também analisamos as principais habilidades obtidas pelos estudantes após a refatoração dos *smells*.

4. Metodologia

4.1. Objetivo e Questões de Pesquisa

Este artigo relata um estudo sobre o ensino da refatoração de *test smells*, realizado com os alunos das turmas de Manutenção de Software e Qualidade de Software da UFC - Campus Quixadá, durante o semestre letivo 2022.2. Nós investigamos as principais habilidades adquiridas pelos estudantes durante a refatoração dos *test smells*, bem como a dificuldade de refatoração dos *test smells* pelos alunos. As questões de pesquisa (QP_s) são apresentadas a seguir.

QP_1: Quais as principais habilidades adquiridas pelos estudantes durante a refatoração dos *test smells*? Esta questão visa verificar e documentar as principais habilidades adquiridas pelos estudantes durante a refatoração dos *test smells*. Ao responder **QP_1**, podemos identificar uma nova categoria de benefícios relacionados a refatoração dos *test smells*, tendo em vista que além da melhoria na qualidade do código de teste, a refatoração de tais *smells* também pode acarretar no desenvolvimento de habilidades pessoais e interpessoais por parte dos responsáveis pela refatoração.

QP_2: Quais os *test smells* que os alunos tiveram mais dificuldade para refatorar? Esta questão visa analisar através do trabalho prático aplicado nas disciplinas, os *test smells* mais difíceis de se refatorar, de acordo com os alunos. Ao responder **QP_2**, podemos examinar os *test smells* as percepções dos alunos, coletadas através da técnica do diário, e a partir disso, identificar relações com as informações referentes aos *test smells* mais difíceis de se refatorar. Através dessas informações, é possível definir novas estratégias de ensino, com ênfase nos *test smells* mais difíceis de se refatorar, e consequentemente, melhorar a acertividade na refatoração de tais *smells*.

4.2. Etapas do estudo

A Figura 1 exibe as etapas necessárias para a realização do estudo.

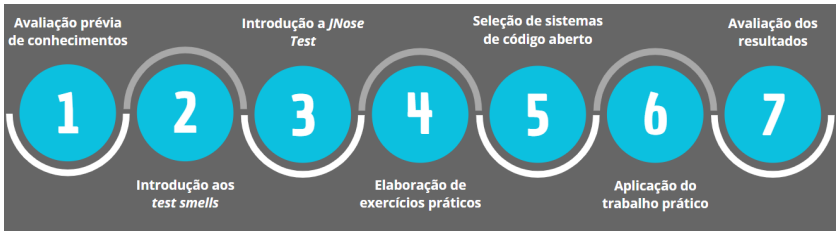

Figura 1. Etapas do estudo

Etapa 1: Avaliação prévia de conhecimentos. A primeira etapa do estudo consistiu na elaboração de um questionário com o intuito de verificarmos o nível de conhecimento prévio dos estudantes em assuntos fundamentais para a compreensão e realização do estudo. Questionamos se os alunos já estudaram ou trabalharam com os seguintes conteúdos: *test smells*; refatoração de software; e testes unitários (JUnit 4 / JUnit 5). O questionário foi aplicado de forma através do *Google Forms*. Como este estudo tem foco no ensino da refatoração de *test smells*, coletar o nível de conhecimento prévio dos alunos foi de suma importância, para que pudéssemos nos adequar a necessidade de cada aluno, e tornar o estudo o mais completo possível. Após a aplicação do questionário, coletamos as seguintes informações:

- 96% dos alunos não tinham conhecimento sobre *test smells*.
- Apenas 53% dos alunos já implementaram testes unitários.
- Apenas 20% dos alunos já refatoraram algum tipo de *test smell*, consciente ou inconscientemente.
- Todos os alunos já possuíam conhecimentos prévios sobre refatoração de software.
- 84% dos alunos já possuíam conhecimentos sobre JUnit 4 / JUnit 5.

Etapa 2: Introdução aos *test smells*. Após a aplicação do questionário e coleta dos dados, seguimos para o planejamento e elaboração das aulas teóricas, ministradas para as turmas de Manutenção de Software e Qualidade de Software. As reuniões foram inicialmente realizadas de forma presencial, e posteriormente de forma remota. O objetivo das aula foi introduzir aos estudantes os principais conceitos relacionados aos *test smells*, que englobam aspectos como definição, riscos a qualidade do código e estratégias adequadas para a refatoração de tais *smells*. O conteúdo das aulas teve ênfase nos *test smells Assertion Roulette, Eager Test, Duplicate Assert, Ignored Test, Magic Number Test, Lazy Test, Constructor Inicialization, Print Statment e Unknown Test*, devido ao fato de o estudo ser baseado exclusivamente na refatoração desses tipos de *test smells*.

Etapa 3: Introdução a *JNose Test*. Após a conclusão das aulas de introdução aos *test smells*, ministramos um treinamento com os alunos, com o intuito de apresentar a ferramenta *JNose Test* [1] [Virgínio et al. 2020, Virgínio et al. 2021], responsável pela detecção dos *test smells* abordados neste estudo. A ferramenta *JNose* coleta *test smells* na linguagem de programação Java e detecta 21 tipos de *test smells* no *framework JUnit*. A *JNose* fornece aos desenvolvedores de software uma interface visual avançada de aspectos relacionados ao código de teste, alternando de indicadores computáveis estaticamente (como atributos de qualidade e *test smells*) para medidas dinâmicas (como cobertura de código e indicadores).

[1] `https://github.com/arieslab/jnose`

Etapa 4: Elaboração de exercícios práticos. Após a realização das aulas de introdução aos *test smells* e do treinamento de detecção de *test smells* via através da *JNose Test*, elaboramos um conjunto de exercícios práticos, que abordaram desde a detecção até a refatoração de todos os tipos de *smells* abordados no estudo. Deste modo, os alunos foram estimulados a identificar os *test smells* através da *JNose Test* e implementar as soluções que lhes foram apresentadas durante as aulas de introdução aos *test smells*.

Etapa 5: Seleção de sistemas de código aberto para detecção de *test smells*. Esta etapa do estudo consiste na seleção de um conjunto de sistemas de software candidatos. Para executar esta etapa, consideramos o conjunto de dados de [Pecorelli et al. 2021], que engloba um conjunto de sistemas de software de código aberto do GitHub, com diversos tipos de *test smells*. A partir deste trabalho, selecionamos dezoito sistemas de software, devido ao fato de tais sistemas possuírem a maior quantidade de *test smells* e de tipos diferentes de *test smells*.

Etapa 6: Aplicação do trabalho prático. Após a seleção dos sistemas com a presença de *test smells*, demos início ao trabalho prático das disciplinas. Solicitamos aos alunos para que se dividissem em equipes com no máximo três integrantes. Ao todo, incluindo as duas turmas, foram formadas 14 equipes, totalizando 30 alunos. Após a formação das equipes, iniciamos a divisão dos *test smells*. Cada equipe ficou responsável pela refatoração de 5 exemplos de cada um dos 9 tipos de *test smells* que foram selecionados para o trabalho. Deste modo, 630 exemplos de *test smells* foram disponibilizados para refatoração.

Para auxiliar na atividade de refatoração, fornecemos aos estudantes uma lista de classes e métodos para identificação dos *test smells* na etapa de detecção. Além disso, solicitamos as equipes para que criassem um repositório no *Github* para acompanharmos o andamento das refatorações com mais precisão. Realizamos duas reuniões semanais para verificar o andamento das refatorações e verificar se os alunos enfrentaram alguma dificuldade. Para manter a organização do experimento, cada estudante foi responsável por criar uma *branch* com todas as operações realizadas para refatorar os *test smells*. Avaliamos cada *branch* para constatar se os *test smells* foram refatorados corretamente de acordo com os procedimentos apresentados na etapa de introdução aos *test smells*.

Além disso, contamos com a técnica do diário para documentar as percepções dos alunos sobre a refatoração dos *test smells*. A técnica do Diário consiste em um método de coleta de dados onde os participantes registram em um formulário suas atividades diárias sobre algum evento que os afetou positiva ou negativamente. Essa técnica é uma forma de entender o comportamento dos participantes, reduzindo o impacto dos pesquisadores [França et al. 2020].

Todas os dados da prática de refatoração dos *test smells* estão disponíveis no repositório [2].

Etapa 7: Avaliação dos resultados. Revisamos as *branchs* de todos os alunos para verificarmos se os *test smells* foram refatorados corretamente. Para tal, levamos em consideração: (i) a técnica de refatoração utilizada para remover o *test smell*; e (ii) a avaliação da JNose para averiguar se o *test smell* foi realmente removido do sistema. Em conjunto com a verificação da refatoração dos *test smells*, investigamos também as

[2] `https://github.com/leanresearchlab/WEI-2023`

informações contidas nos diários preenchidos pelos alunos, a fim de identificarmos os *test smells* mais difíceis de se refatorar. Além disso, também analisamos as habilidades pessoais e interpessoais adquiridas pelos estudantes durante / após a refatoração dos *test smells*, a fim de identificarmos os benefícios da prática de refatoração de tais anomalias. Para realizarmos a coleta de habilidades, aplicamos um novo questionário com os estudantes. O questionário abordou aspectos relacionados a: (i) habilidades adquiridas durante / após a refatoração dos *smells*; (ii) os tipos de *test smells* mais difíceis de se refatorar; (iii) os *test smells* mais prejudiciais para o código de teste, sob a perspectiva do aluno; e (iv) as dificuldades encontradas durante a refatoração dos *test smells*. A remoção completa dos *test smells* por meio da refatoração manual levou cerca de dois meses.

5. Resultados

5.1. Habilidades adquiridas através da refatoração dos *test smells* ($\mathbf{QP_1}$)

Resolvemos a $\mathbf{QP_1}$ avaliando as respostas dos alunos geradas a partir de um questionário *online*. As habilidades foram extraídas da revisão sistemática sobre habilidades de engenheiros de software, realizada por [Matturro et al. 2019]. A partir disso, realizamos uma análise qualitativa através das respostas coletadas, e obtivemos 9 categorias principais de habilidades pessoais e interpessoais adquiridas pelos alunos após refatoração dos *test smells*, que são apresentadas na Tabela 1. A primeira coluna apresenta todas as categorias, a segunda coluna apresenta a descrição das competências e a terceira contém o número de alunos que adquiriram essas competências.

Tabela 1. Habilidades adquiridas pelos estudantes após a refatoração dos *test smells* [Matturro et al. 2019]

Habilidade	Descrição	Total
Habilidades analíticas	A capacidade de entender e explicar cada parte de um todo, de conhecer melhor que a natureza, funções, causas, entre outros	16
Resolução de problemas	A capacidade de compreender, articular e resolver problemas complexos	20
Pensamento crítico	A capacidade de determinar cuidadosa e deliberadamente aceita, refutação ou suspensão do julgamento sobre uma determinada informação	20
Gerenciamento de mudanças	A capacidade de propor e/ou realizar qualquer ação sem a necessidade de que outros venham pedir ou dizer	7
Trabalho em equipe	Capacidade de um indivíduo que é bom em trabalhar em estreita colaboração com outras pessoas	14
Tomada de decisões	A capacidade de tomar decisões sensatas com base nas informações disponíveis	12
Orientação a resultados	A capacidade de alcançar e/ou exceder metas e/ou objetivos de vendas	3
Metodismo	A capacidade de usar um conjunto de etapas, ordenadamente, organizadas, definidas por métodos (técnicas) para resolver uma questão ou problema específico	3
Criatividade	É a capacidade de criar, imaginar ou produzir algo novo e diferente	8

Analisando as informações contidas na Tabela 1, percebe-se que a resolução de problemas (20), pensamento crítico (20) e habilidades analíticas (16), foram as mais citadas pelos alunos no questionário. Portanto, infere-se que a atividade de refatoração de *test smells* impactou positivamente os alunos na aquisição de novas habilidades pessoais, que podem ser úteis em diversas situações, tanto acadêmicas, quanto profissionais.

Descoberta 1: O trabalho prático sobre refatoração de *test smells* forneceu novas habilidades para os alunos, especialmente para resolução de problemas, pensamento crítico e habilidades analíticas.

Na Tabela 1, é possível notar que os alunos também melhoraram suas habilidades interpessoais. Considerando que habilidades como trabalho em equipe (12), tomada de

decisão (10), criatividade (8) e gerenciamento a mudanças (7). Portanto, pode-se perceber que a atividade de refatoração de *test smells* também melhorou as habilidades interpessoais dos alunos, tendo em vista que vários tipos de habilidades foram listadas pelos estudantes.

Descoberta 2: A refatoração de *test smells* também se mostrou benéfica para o aprimoramento de habilidades interpessoais dos alunos.

Implicações de $\mathbf{QP}_1$. Nossas descobertas indicam que a aplicação do trabalho prático sobre refatoração de *test smells* aos alunos foi benéfica. Considerando que as habilidades pessoais e interpessoais melhoraram após a refatoração dos *test smells*. Tais benefícios ressaltam a importância de ensinar sobre *test smells* e suas devidas estratégias de refatoração em cursos de computação. Tendo em vista que é um tema de grande relevância para a comunidade acadêmica, principalmente pelo aprimoramento das habilidades dos alunos na indústria.

5.2. *Test smells* mais difíceis de se refatorar, segundo os alunos ($\mathbf{QP_2}$)

Discutimos $\mathbf{QP}_2$ analisando as respostas obtidas através do questionário aplicado após a refatoração dos *smells*, para identificarmos os tipos de *test smells* mais difíceis de se refatorar de acordo com os alunos. Além disso, também investigamos as informações coletadas por meio da técnica do diário, onde cada aluno documentou suas impressões sobre os *test smells*. A Tabela 2 apresenta os *test smells* mais difíceis de se refatorar de acordo com os alunos. A primeira coluna apresenta os tipos de *test smells* e a segunda coluna contém a quantidade de vezes que o *test smell* foi considerado difícil para se refatorar.

Tabela 2. *Test smells* mais difíceis de se refatorar na perspectiva dos alunos

Test smell	Quantidade
Eager Test	16
Duplicate Asserti	8
Assertion Roulette	1
Ignored Test	0
Lazy Test	7
Magic Number Test	1
Constructor Inicialization	4
Unknown Test	5
Print Statment	0

Através da Tabela 2, é possível notar que os *test smells Eager Test (16), Duplicate Assert (8) e Lazy Test (7)* foram considerados os mais difíceis de se refatorar. É possível notar que tais *test smells* necessitam de extração de métodos (*Eager Test* e *Duplicate Assert*) ou junção de métodos (*Lazy Test*). Logo, isso infere que os *test smells* que necessitam das técnicas de refatoração *extract method* ou *inline resource* para serem removidos, consequentemente serão mais difíceis de se refatorar. Alguns relatos dos alunos fortalecem essa afirmação:

Aluno 5:"O test smell Eager Test foi o mais difícil de se refatorar. Tive problemas para fazer a extração do método."

Aluno 11:"Tive dificuldades para aplicar a técnica de refatoração, e consequentemente remover o test smell Duplicate Assert."

Aluno 19:"Dentre os smells repassados, o mais difícil de se refatorar foi o Eager Test. Muito devido a sua forma de ser refatorado."

Aluno 23:"Tive dificuldades para remover o test smell Lazy Test. Pois tive dificuldade em aplicar a técnica de refatoração Inline Resource."

Descoberta 3: *Test smells* que necessitam das técnicas de refatoração *Extract method* ou *Inline resource* para serem removidos são mais difíceis de se refatorar.

Por outro lado, ainda sobre a Tabela 2, é possível notar que os *test smells Ignored Test*, *Print Statment*, *Assertion Roulette* e *Magic Number Test* foram considerados os menos difíceis de se refatorar. Isso se deve ao fato de que todos esses tipos de *test smells* não necessitam de muita alteração de código para que a refatoração seja concluída. Especialmente os *smells Assertion Roulette* e *Magic Number Test*, que podem ser refatorados com a alteração de apenas uma linha de código. Alguns relatos dos alunos vão fortalecem a afirmação de que os *test smells* mais fáceis de se refatorar são os que demandam uma menor quantidade de alteração de código:

Aluno 2:"Não tive nenhuma dificuldade para refatorar os smells Assertion Roulette e Magic Number Test."

Aluno 16:"Dentre os smells repassados, considero o smell Print Statment o de refatoração mais simples e prática."

Aluno 29:"Assertion Roulette e Magic Number Test foram simples de refatorar, pois foram necessárias pequenas alterações de código para a conclusão da refatoração."

Descoberta 4: *Test smells* que necessitam de pouca alteração de código são os mais fáceis de se refatorar. Especialmente o *Assertion Roulette* e *Magic Number Test*, que podem ser refatorados com a alteração de apenas uma linha de código.

Implicações de QP_2. Nossos achados apontam que os *test smells* que dependem das técnicas de refatoração *extract method* ou *inline resource* (*Eager Test, Duplicate Assert* e *Lazy Test*) para serem removidos, foram considerados pelos alunos como os *test smells* mais difíceis de se refatorar. Por outro lado, os *test smells* que necessitam de pouca alteração no código para serem refatorados (*Ignored Test*, *Print Statment*, *Assertion Roulette* e *Magic Number Test*) foram considerados os mais fáceis de se refatorar. Devido aos resultados obtidos, é de suma importância que a prática do ensino de refatoração de *test smells* seja mais completa no que diz respeito as técnicas de refatoração utilizadas para a refatoração dos *test smells*, sendo a utilização dessas técnicas uma das dificuldades por parte dos alunos.

6. Limitações do estudo

Como limitações do estudo, tem-se: (i) falta de experiência de alguns alunos, minimizada através das aulas introdutórias; (ii) a utilização de apenas uma ferramenta para detecção dos *test smells*; (iii) não foram investigados os *test smells* que fazem a utilização de recursos externos, pelo fato do *recall* da ferramenta utilizada para a detecção não ser efetiva

para esse tipo de *test smell*; e (iv) a estratégia para a refatoração dos *test smells* foi decidida pelos próprios alunos, porém, validamos todas as refatorações através da JNose Test.

7. Conclusão

Neste estudo, apresentamos um relato de experiência no ensino dos conceitos de detecção, refatoração e remoção dos *test smells*, analisando a qualidade do código de testes. A metodologia de ensino foi aplicada nas disciplinas de Qualidade de Software e Manutenção de Software do curso de Engenharia de Software da UFC - Campus Quixadá com 30 alunos no semestre de 2022.2. Para este artigo, nós analisamos: (i) as principais habilidades adquiridas pelos estudantes durante a refatoração de 630 exemplos de *test smells* e (ii) os *test smells* mais difíceis de se refatorar na perspectiva dos alunos.

Nossos principais achados foram: (i) o trabalho prático forneceu novas habilidades para os alunos, especialmente para resolução de problemas, pensamento crítico e habilidades analíticas; (ii) a refatoração dos *test smells* também se mostrou benéfica para o aprimoramento das habilidades interpessoais dos alunos; (iii) os *test smells* que necessitam das técnicas de refatoração *Extract Method* ou *Inline Resource* para serem removidos, são os mais difíceis de se refatorar; e (iv) os *test smells Assertion Roulette* e *Magic Number Test* são mais fáceis de refatorar por necessitar de pouca alteração de código.

Com base nos resultados obtidos, desenvolvemos as seguintes recomendações para o ensino da refatoração de *test smells*: (i) realização de mais atividades práticas de detecção e refatoração de *test smells*; (ii) maior ênfase no ensino das técnicas de refatoração que necessitam de extração de código, tendo em vista que estas foram as principais responsáveis pelas dificuldades relatadas pelos estudantes; e (iii) maior ênfase no ensino dos *test smells* que demandam mais esforço para serem refatorados.

Como trabalhos futuros, pretende-se: (i) replicar o estudo com um maior número de projetos e novos *test smells*; (ii) usar outras ferramentas para identificar *test smells*; (iii) analisar sistemas desenvolvidos em outras linguagens de programação; (iv) analisar a prática do ensino de co-ocorrências de *test smells*; e (v) avaliar a capacidade dos alunos em detectar manualmente os *test smells*.

8. Agradecimentos

Este estudo foi financiado pela Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP-CE).

Referências

Alizadeh, V., Kessentini, M., Mkaouer, M. W., Ó Cinnéide, M., Ouni, A., and Cai, Y. (2020). An interactive and dynamic search-based approach to software refactoring recommendations. *IEEE Transactions on Software Engineering*, 46(9):932–961.

Aljedaani, W., Mkaouer, M. W., Peruma, A., and Ludi, S. (2023). Do the test smells assertion roulette and eager test impact students' troubleshooting and debugging capabilities? *arXiv preprint arXiv:2303.04234*.

Aniche, M., Hermans, F., and van Deursen, A. (2019a). Pragmatic software testing education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science*

Education, SIGCSE '19, page 414–420, New York, NY, USA. Association for Computing Machinery.

Aniche, M., Hermans, F., and Van Deursen, A. (2019b). Pragmatic software testing education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 414–420.

Bai, G. R., Smith, J., and Stolee, K. T. (2021). How students unit test: Perceptions, practices, and pitfalls. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, pages 248–254.

Bavota, G., Qusef, A., Oliveto, R., De Lucia, A., and Binkley, D. (2012). An empirical analysis of the distribution of unit test smells and their impact on software maintenance. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 56–65.

Beller, M., Gousios, G., Panichella, A., Proksch, S., Amann, S., and Zaidman, A. (2019). Developer testing in the ide: Patterns, beliefs, and behavior. *IEEE Transactions on Software Engineering*, 45(3):261–284.

Beller, M., Gousios, G., Panichella, A., and Zaidman, A. (2015). When, how, and why developers (do not) test in their ides. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, page 179–190, New York, NY, USA. Association for Computing Machinery.

Berner, S., Weber, R., and Keller, R. (2005). Observations and lessons learned from automated testing. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 571–579.

Buffardi, K. and Aguirre-Ayala, J. (2021). Unit test smells and accuracy of software engineering student test suites. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, pages 234–240.

Campos, D., Rocha, L., and Machado, I. (2021). Developers perception on the severity of test smells: an empirical study. *XXIV Ibero-American Conference on Software Engineering*.

Candea, G., Bucur, S., and Zamfir, C. (2010). Automated software testing as a service. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, page 155–160, New York, NY, USA. Association for Computing Machinery.

Fowler, M. (1999). *Refactoring: improving the design of existing code*. ilustrated edition.

França, C., da Silva, F. Q. B., and Sharp, H. (2020). Motivation and satisfaction of software engineers. *IEEE Transactions on Software Engineering*, 46(2):118–140.

Garousi, V. and Küçük, B. (2018). Smells in software test code: A survey of knowledge in industry and academia. *Journal of Systems and Software*, 138:52–81.

Garousi, V., Rainer, A., Lauvås Jr, P., and Arcuri, A. (2020). Software-testing education: A systematic literature mapping. *Journal of Systems and Software*, 165:110570.

Kim, D. J. (2020). An empirical study on the evolution of test smell. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, ICSE '20, page 149–151, New York, NY, USA. Association for Computing Machinery.

Matturro, G., Raschetti, F., and Fontán, C. (2019). A systematic mapping study on soft skills in software engineering. *J. Univers. Comput. Sci.*, 25(1):16–41.

Myers, G. J., Sandler, C., and Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

Orso, A. and Rothermel, G. (2014). Software testing: A research travelogue (2000–2014). In *Future of Software Engineering Proceedings*, FOSE 2014, page 117–132, New York, NY, USA. Association for Computing Machinery.

Paixão, M., Uchôa, A., Bibiano, A. C., Oliveira, D., Garcia, A., Krinke, J., and Arvonio, E. (2020). *Behind the Intents: An In-Depth Empirical Study on Software Refactoring in Modern Code Review*, page 125–136. Association for Computing Machinery, New York, NY, USA.

Palomba, F., Zaidman, A., and De Lucia, A. (2018). Automatic test smell detection using information retrieval techniques. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 311–322.

Pecorelli, F., Palomba, F., and De Lucia, A. (2021). The relation of test-related factors to software quality: A case study on apache systems. *Empirical Software Engineering*, 26(2):1–42.

Peruma, A., Almalki, K., Newman, C. D., Mkaouer, M. W., Ouni, A., and Palomba, F. (2019). On the distribution of test smells in open source android applications: An exploratory study. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, CASCON '19, page 193–202, USA. IBM Corp.

Sjøberg, D. I., Yamashita, A., Anda, B. C., Mockus, A., and Dybå, T. (2013). Quantifying the effect of code smells on maintenance effort. *IEEE Transactions on Software Engineering*, 39(8):1144–1156.

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

Van Deursen, A., Moonen, L., Van Den Bergh, A., and Kok, G. (2001). Refactoring test code. In *Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP2001)*, pages 92–95. Citeseer.

Virgínio, T., Martins, L., Rocha, L., Santana, R., Cruz, A., Costa, H., and Machado, I. (2020). Jnose: Java test smell detector. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, SBES '20, page 564–569, New York, NY, USA. Association for Computing Machinery.

Virgínio, T., Martins, L., Santana, R., Cruz, A., Rocha, L., Costa, H., and Machado, I. (2021). On the test smells detection: an empirical study on the jnose test accuracy. *Journal of Software Engineering Research and Development*, 9:8–1.

Yusifoğlu, V. G., Amannejad, Y., and Can, A. B. (2015). Software test-code engineering: A systematic mapping. *Information and Software Technology*, 58:123–147.