Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Test Smells: Uma Perspectiva de Desenvolvedores da Indústria

Gabriel Crespo de Souza

Orientador
Prof. Dr. Rodrigo Bonifácio de Almeida

Brasília
2023

Test Smells: Uma Perspectiva de Desenvolvedores da Indústria

Gabriel Crespo de Souza[1], Rodrigo Bonifácio de Almeida[2], Walter Lucas[2]

[1] Departamento de Ciência da Computação
Universidade de Brasília (UnB)
Brasília, Distrito Federal, Brasil

gabriel.crespo@aluno.unb.br, rbonifacio@unb.br, walter.mendonca@aluno.unb.br

Resumo. *Test Smells são uma má prática de programação que ocorrem durante o desenvolvimento de testes de unidade, bem como uma potencial ameaça ao código-fonte de um projeto, pois mascaram eventuais inconsistências que nele possam estar presentes, diminuindo drasticamente a sua qualidade. Como uma má escolha, Test Smells tendem a depreciar a suíte de testes de um projeto, tornando complexo o entendimento e a manutenção dos códigos de teste. Alguns estudos recentes têm buscado entender se há a percepção por parte da indústria sobre o termo Test Smells e suas implicações. Esse estudo tem como objetivo principal descobrir qual a visão de desenvolvedores de software que atuam na indústria sobre Test Smells. Nesse estudo foram coletados 215 exemplos de 7 tipos de Test Smells existentes na literatura e presentes em um projeto produzido pela indústria. Os resultados indicam que desenvolvedores com uma considerável experiência podem reconhecer e até mesmo solucionar Test Smells, sem conhecimento prévio do termo. Outra importante descoberta foi que desenvolvedores consideram importante a correção de Test Smells por meio de ferramentas auxiliares, trazendo mais agilidade e qualidade aos projetos em que atuam.*

1. Introdução

A qualidade de um *software* depende inteiramente de um processo bem definido de teste. Todo *software* de grande porte, comercial ou de código aberto, atualmente incluem suítes de testes que verificam a sua funcionalidade [Bavota et al. 2012a]. Testes manuais ou automatizados são as maneiras às quais são executados os testes de *software* [Garousi and Küçük 2018]. Quando executados de forma manual, existem procedimentos que necessitam de um grande custo e esforço [Campos et al. 2021]. Em contrapartida, testes automatizados tornam as execuções dos testes previsíveis, passíveis de repetição e eficientes, mas como todo processo de desenvolvimento, as automações precisam passar por avaliações e manutenções, de modo a garantir sua qualidade [Bavota et al. 2012a].

Projetar o desenvolvimento de uma suíte de testes automatizados não é uma tarefa tão simples quanto parece [Campos et al. 2021]. Durante essa atividade, más decisões podem ser tomadas, levando os desenvolvedores a não seguirem o modo mais apropriado para o desenvolvimento dos códigos de teste. Na maior parte do tempo, os desenvolvedores têm limitações quando precisam escrever os testes da aplicação, seja por falta de tempo ou por falta de investimento, o que os leva a

desenvolver testes que fujam do padrão, isto é, testes com anomalias. Anomalias essas que são identificadas por meio do termo *Test Smells* [Junior et al. 2021].

Embora não sejam vistos como um problema em si, *Test Smells* sugerem um potencial trecho de código no teste que pode ser capaz de levá-lo à falha [Bavota et al. 2015]. Assim como qualquer processo de desenvolvimento, os testes de um *software* devem seguir determinados padrões que tornem claro o entendimento e fácil a manuntenção. Contrariando essa afirmação, *Test Smells* indicam um pobre projeto de implementação, e que podem afetar a manutenibilidade e o entendimento de um código de teste [Santana et al. 2021].

Muitos estudos têm buscado entender qual o tamanho da percepção por parte dos desenvolvedores em relação ao conceito de *Test Smells* e suas implicações[Campos et al. 2021, Junior et al. 2021]. O principal objetivo desse estudo é estender e enriquecer trabalhos prévios, buscando entender qual a visão dos desenvolvedores que atuam na indústria sobre *Test Smells*, se consideram o tema relevante nos projetos que desenvolvem no dia-a-dia, se de fato enxergam *Test Smells* como um potencial problema aos testes do projeto, bem como se eles enxergam algum valor em ferramentas que auxiliem em identificar possíveis *Test Smells* disfarçados e, até mesmo, os auxiliem com possíveis correções.

O estudo foi conduzido em partes. Na primeira parte, uma companhia na indústria foi consultada, em busca de um projeto que pudesse ser disponibilizado, de modo a ser utilizado como base para a condução desse estudo. Na segunda parte, foram selecionados 4 desenvolvedores com larga experiência, e que fossem colaboradores da empresa e do projeto, a fim de coletar, analisar e entender qual a percepção deles sobre *Test Smells*. Por fim, os dados coletados foram analisados. A ferramenta *JNose Test* [Santana et al. 2021] foi utilizada como apoio no projeto selecionado para a identificação dos *Test Smells*.

2. Fundamentos e Trabalhos Relacionados

Test Smells podem ser definidos como sintomas de presença de más escolhas em um projeto ou em uma implementação de códigos de teste [Palomba et al. 2016]. Ou seja, sua presença indica que desenvolvedores falharam ao seguir bons principios de desenvolvimento [Bavota et al. 2015]. Estudos recentes têm mostrado que a existência de *Test Smells* nos códigos de testes de um projeto pode reduzir de forma considerável o seu entendimento, especialmente quando muitos deles são combinados [Bavota et al. 2012b]. Alguns artigos são pioneiros no assunto e abordam um catálogo de *Test Smells*, bem como algumas possíveis refatorações [Campos et al. 2021, Peruma 2018, Van Deursen et al. 2001].

Nesse estudo, foram analisados os 4 tipos de *Test Smells* de maior frequência encontrados pela ferramenta, no caso: *Assertion Roulette*, *Eager Test*, *Magic Number* e *Lazy Test*.

Assertion Roulette: ocorre quando um método de teste contém múltiplos comandos de asserção sem qualquer explicação ou parâmetro no método de asserção [Campos et al. 2021]. O trecho de código 1 ilustra esse tipo de *Test Smells*:

Listing 1. Exemplo do tipo de *Test Smells* conhecido como *Assertion Roulette*.

```
@Test
public void FizzBuzzUpTo15() {

    FizzBuzz fb = new FizzBuzz();
    List<String> result = fb.fizzBuzz(15);

    Assertions.assertEquals("11", result.get(0));
    Assertions.assertEquals("Fizz", result.get(2));
    Assertions.assertEquals("Buzz", result.get(4));
    Assertions.assertEquals("FizzBuzz", result.get(14));

}
```

Eager Test: ocorre quando um método de teste contém diversas chamadas para diferentes métodos de um objeto [Campos et al. 2021]. O trecho de código 2 ilustra esse tipo de *Test Smells*:

Listing 2. Exemplo do tipo de *Test Smells* conhecido como *Eager Test*.

```
@Test
public void FizzBuzzUpTo15() {

    FizzBuzz fb = new FizzBuzz();
    List<String> result = fb.fizzBuzz(15);
    boolean b = fb.isDivideByThreeAndFive(15);

    Assertions.assertEquals("FizzBuzz", result.get(14));
    Assertions.assertTrue(b);

}
```

Magic Number: ocorre quando um comando de asserção contém valores literais como parêmetros [Peruma 2018]. O trecho de código 3 ilustra esse tipo de *Test Smells*:

Listing 3. Exemplo do tipo de *Test Smells* conhecido como *Magic Number*.

```
@Test
public void FizzBuzzShouldReturnLiteralWhenNotDivisible() {

    FizzBuzz fb = new FizzBuzz();
    List<String> result = fb.fizzBuzz(15);

    Assertions.assertEquals(1, Integer.parseInt(result.get(0)));

}
```

Lazy Test: ocorre quando múltiplos metódos de teste invocam o mesmo método de um objeto [Campos et al. 2021]. O trecho de código 4 ilustra esse tipo de *Test Smells*:

Listing 4. Exemplo do tipo de *Test Smells* conhecido como *Lazy Test*.

```
@Test
public void FizzBuzzUpTo15() {

    FizzBuzz fb = new FizzBuzz();
    List<String> result = fb.fizzBuzz(15);

    Assertions.assertEquals("Fizz", result.get(2));

}

@Test
public void FizzBuzzUpTo30() {

    FizzBuzz fb = new FizzBuzz();
    List<String> result = fb.fizzBuzz(30);

    Assertions.assertEquals("Fizz", result.get(2));

}
```

Conforme abordado em [Junior et al. 2021], por Silva Junior, *Test Smells* podem ter um impacto negativo na qualidade e na manuntenção de códigos de teste, podendo também prejudicar as atividades de teste. Portanto, seu estudo buscou analisar a frequência em que são usados exemplos de *Test Smells* por desenvolvedores de *software*. Os resultados indicam que mesmo desenvolvendo códigos de teste seguindo os melhores padrões de projeto, os desenvolvedores introduzem *Test Smells* nos projetos em que atuam. Outro importante resultado encontrado em [Junior et al. 2021] é que a experiência de um desenvolvedor não está diretamente ligada à introdução de *Test Smells* em códigos de teste. Esse estudo teve por objetivo mostrar que a experiência de um desenvolvedor pode também influenciar na identificação e correção de casos de *Test Smells*, mesmo que não haja conhecimento prévio sobre o conceito.

Estudos recentes têm mostrado que a indústria não tem consciência do termo *Test Smells*, enquanto os engenheiros de *software* comumente encontram obstáculos para manter seus códigos de teste [Santana et al. 2021]. O estudo [Junior et al. 2020], por Ivan Machado, apresenta uma investigação sobre a criação e manutenção de testes por meio da percepção de profissionais de teste, e demonstram resultados sobre a preferência dos profissionais em escrever e manter testes de forma manual, bem como a sua consciência sobre o conceito de *Test Smells* e suas severidades, baseado nos exemplos apresentados. Uma das intenções desse estudo foi buscar entender a percepção de desenvolvedores de *software* que atuam na indústria sobre o conceito de *Test Smells* e sua importância nos projetos em que atuam.

Ivan Machado em [Campos et al. 2021] mostra que testes de unidade são parte essencial para o ciclo de vida de um *software*, trazendo mais confiança e auxiliando os desenvolvedores a encontrar possíveis erros em suas implementações, porém, se forem tomadas más decisões de projeto e desenvolvimento, então *Test Smells* são facilmente inseridos. Portanto, seu estudo buscou avaliar a percepção por parte de desenvolvedores de projetos de código aberto sobre a severidade dos

Test Smells. Os resultados mostram que mesmo a percepção deles sobre a severidade de *Test Smells* sendo baixa, eles acreditam que podem impactar negativamente no projeto, principalmente em relação a manutenção. Esse estudo apresenta uma visão similar, porém pelo ponto de vista de desenvolvedores de *software* que atuam na indústria e pela percepção alta sobre a severidade da presença de *Test Smells* nos códigos de teste.

3. Metodologia da Pesquisa

O propósito geral desse estudo é investigar qual a visão de desenvolvedores de *software* que atuam na indústria sobre o tema *Test Smells*, e logo após, responder as questões relacionadas à pesquisa apresentadas na seção 3.1. Para tal finalidade, a pesquisa foi conduzida em duas fases, ambas utilizando uma abordagem multimétodologica.

Na primeira fase foi realizada uma avaliação quantitativa em um projeto da indústria, coletando os principais dados sobre o projeto. Posteriormente foram identificados todos os casos de *Test Smells* presentes na suíte de testes com o auxílio de uma ferramenta que os identifica, e por fim, foram selecionados os 4 tipos de *Test Smells* que ocorriam com maior frequência no projeto, e de forma aleatória, foram coletados 2 exemplos de cada um deles.

Na segunda fase foi realizada uma avaliação qualitativa por meio de um grupo focal com desenvolvedores que atuam ou atuaram diretamente no projeto utilizado como base desse estudo. Durante o grupo focal, uma discussão foi proposta em cima dos exemplos apresentados, o termo *Test Smells* foi introduzido e cada exemplo explicado. Por fim, os participantes responderam à questões que buscavam apoiar o entendimento sobre suas visões do conceito de *Test Smells*.

3.1. Questões da Pesquisa

Foram investigadas as seguintes questões de pesquisa no estudo.

(Q1) *Os participantes têm consciência sobre o termo "Test Smells"?*
(Q2) *O quão relevante esse tema é nos projetos em que atuam?*
(Q3) *Existe o interesse na resolução dos Test Smells apresentados?*
(Q4) *Ferramentas que auxiliem a identificação ou a correção de Test Smells têm relevância para a indústria?*

Esse estudo foi conduzido por meio de uma abordagem interativa. As respostas para cada uma das perguntas, após a introdução do termo *Test Smells* e a explicação dos exemplos apresentados, nos mostraram, por exemplo, a influência na percepção dos participantes sobre o tanto que a presença de *Test Smells* nos códigos de teste de um projeto pode afetar a sua qualidade.

3.2. Métricas do Estudo Quantitativo

As métricas do projeto foram coletadas com o auxílio de duas ferramentas: *Statistics*[Topinka 2009] e *JNose Test*[Santana et al. 2021].

Statistics é um *plugin* desenvolvido para ser executado no ambiente de desenvolvimento integrado *IntelliJ IDEA*[JetBrains 2001]. Sua finalidade é mostrar

algumas estatísticas sobre um projeto específico, tais como a quantidade linhas de código que um projeto contém no total, a quantidade de linhas de código que um pacote contém, a quantidade de linhas de código que um único arquivo contém, etc. Com esse *plugin* foi possível obter resultados sobre quantas linhas de código o projeto tinha no total, e desse total, quantas linhas de código eram da aplicação e quantas eram relativas aos testes.

JNose Test é uma ferramenta que facilita a identificação de *Test Smells* ao analisar os códigos de teste de um projeto Java que utilizam *Maven*[van Zyl 2004] como gerenciador de pacote e *JUnit*[Kent Beck 1997] como *framework* de testes. Atualmente ela detecta 21 tipos de *Test Smells*. Por meio de um conjunto de regras, a ferramenta identifica e quantifica os tipos de *Test Smells*. Com auxílio dessa ferramenta, foi possível quantificar quantas classes de testes do projeto possuíam *Test Smells*, quantos eram os *Test Smells* em cada classe, quantos *Test Smells* o projeto inteiro possuía, bem como a quantidade total de cada de tipo de *Test Smells*, e desse total, o quanto de cada tipo cada classe de teste possuía.

3.3. Conjunto de Dados de Códigos de Teste

A ferramenta *JNose Test*[Santana et al. 2021] foi utilizada para identificar trechos de código candidatos para pesquisa. Essa ferramenta detecta automaticamente *Test Smells* em códigos de teste, coleta métricas de cobertura e tem sido utilizada em pesquisas anteriores sobre *Test Smells*. *JNose Test* se conecta com o diretório do projeto a fim de identificar, classificar e quantificar as ocorrências de *Test Smells* em códigos de testes escritos em Java. A decisão de usar *JNose Test* foi de facilitar a busca por casos reais de presença de *Test Smells*. o A ferramenta possibilita quatro tipos de busca e configuração a fim de detectar *Test Smells*:

- *By ClassTest*: realiza uma busca utilizando a classe de teste como base, e retorna a quantidade de cada tipo de *Test Smells* encontrado em cada classe.
- *By TestSmells*: realiza a busca com base no *Test Smells*, e retorna a qual classe e linha ele foi encontrado.
- *Evolution*: realiza a busca no repositório git do projeto em busca de *Test Smells* em cada commit realizado.
- *Configuration*: possibilita a escolha de quais *Test Smells* se quer realizar a busca. Por padrão, todos estão selecionados.

A partir da possibilidade de coletar os exemplos do projeto, foram selecionados, de forma aleatória, os exemplos. Para o projeto selecionado foram encontrados 215 casos de *Test Smells* em toda a sua suíte de testes. A extração dos exemplos considerou exclusivamente o primeiro tipo de busca da ferramenta, com isso, foi obtido acesso aos dados de cada classe em relação à presença de *Test Smells*. Desse modo, os quatro *Test Smells* de maior frequência no projeto foram escolhidos, e depois foram também selecionados aleatoriamente dois exemplos de cada para serem utilizados.

Dado que os *Test Smells* já haviam sido descobertos e caracterizados pela ferramenta *JNose Test*, foi automatizada a forma com que os exemplos foram coletados. O *script* foi desenvolvido de modo a receber o arquivo CSV gerado pela ferramenta *JNose Test*, depois analisar as informações presentes nesse arquivo, e

retornar o arquivo e a linha do *Test Smells* a ser utilizado como exemplo. Por fim, os exemplos foram examinados manualmente para se ter a certeza que, de fato, caracterizavam um tipo de *Test Smells*. Essas informações ajudaram com a sessão do grupo focal e com as análises.

3.4. Procedimentos do Estudo Qualitativo

A respeito do estudo qualitativo, o estudo foi conduzido por meio da abordagem baseada no trabalho de Kontio, Lehtola e Bragge [Kontio et al. 2004]. Isso é, foi organizado um grupo focal que permitisse os participantes avaliarem os exemplos de *Test Smells* coletados. Foram convidados desenvolvedores profissionais que tivessem colaborado ativamente no desenvolvimento do projeto, e que também tivessem uma longa experiência com a linguagem de programação Java. Ou seja, desenvolvedores de *software* profissionais foram a população de interesse.

Posteriormente, a entrevista foi realizada junto ao grupo focal. A entrevista foi realizada em quatro etapas, uma explicando aos participantes a sua finalidade, o sigilo sobre seus dados e a solicitação de suas permissões para que a conversa fosse gravada. Uma para caracterizar a experiência dos participantes. Uma para entender quais as percepções dos participantes sobre os exemplos de *Test Smells* apresentados. E uma para explicar os conceitos e capturar a percepção dos participantes sobre a importância desse tema. Logo após obter a permissão de todos para que a entrevista fosse realizada, os desenvolvedores foram questionados sobre alguns dados pessoais, e essas características são apresentadas na tabela 1.

Durante a entrevista, a cada exemplo apresentado, a ideia central era analisar se cada participante enxergava um problema ou não, a medida em que os exemplos iam sendo apresentados. Caso um participante encontrasse alguma inconsistência no exemplo apresentado, então ele era instigado a falar mais sobre o problema que ele conseguiu visualizar.

Logo após passar por todos exemplos e obter a percepção dos desenvolvedores, o conceito de *Test Smells* foi explicado, bem como seus malefícios, e então, cada exemplo foi explicado e o porquê dele caracterizar um tipo de *Test Smells*. Por fim, foram feitos questionamentos que ajudassem a entender se após obterem consciência do significado do termo *Test Smells* e suas implicações, eles consideravam um tema importante nos trabalhos que faziam no dia-a-dia, e se ferramentas que auxiliassem a identificação ou correção de *Test Smells*, de fato, teriam algum valor.

Inicialmente, um piloto foi conduzido com um único desenvolvedor também contribuinte do projeto, com finalidade de avaliar se a metodologia utilizada seria capaz de capturar a opinião dos participantes do grupo focal. Após a condução do piloto, alguns ajustes foram feitos no roteiro previamente pensado para condução do grupo focal.

4. Resultados

Nessa seção serão apresentados os resultados das duas fases do estudo. Primeiramente serão apresentados os resultados da avaliação quantitativa. E logo após,

Id	Cargo	Experiência Java	Experiência Testes Unitários
1	Desenvolvedor Sênior	10 anos	3 anos
2	Desenvolvedor Pleno	4 anos	3 anos
3	Desenvolvedor Pleno	6 anos	3 anos
4	Líder Técnico	13 anos	10 anos

Tabela 1. Caracterização dos participantes da entrevista por seus cargos e níveis de experiência.

	Quantidade	Porcentagem
Total de Linhas de Código	3547	100%
Linhas de Código do Projeto	2425	68,4%
Linhas de Código de Teste	1122	31,6%

Tabela 2. Métricas relativas às quantitidades totais de linhas de código do projeto e de sua súite de testes.

serão apresentados os resultados da avaliação qualitativa, e então serão comparadas as descobertas dos dois estudos.

4.1. Avaliação quantitativa

Foi considerado um projeto Java desenvolvido na indústria durante a avaliação quantitativa. Com o apoio de ferramentas como *JNose Test* e *Statistic*, se fez possível a coleta de métricas importantes sobre o projeto utilizado como base desse estudo. Dentre as métricas coletadas estão a quantidade total de código do projeto, a quantidade de linhas de código relativas ao sistema em si, a quantidade de linhas de código de teste, a quantidade classes relativas à suíte de testes do projeto, bem como a quantidade total de *Test Smells* encontrados na suíte de testes inteira, também diferenciados por seus tipos. As tabelas 2 e 3 resumem os resultados descritos acima. Para esse trabalho foram escolhidos os 4 *Test Smells* mais frequentes nos codígos de testes, que conforme a tabela 3 são: *Assertion Roulette*, *Eager Test*, *Magic Number* e *Lazy Test*.

	Quantidade	Porcentagem (≈)
Total de *Test Smells*	215	100%
Assertion Roulette	125	58%
Magic Number	46	22%
Lazy Test	20	10%
Eager Test	18	8%
Unknown Test	2	1%
Duplicate Assert	2	1%

Tabela 3. Métricas relativas aos tipos de *Test Smells* encontrados no projeto e suas frequências.

4.2. Avaliação qualitativa

Considerando a avaliação qualitativa, 4 participantes com uma considerável experiência com programação utilizando a linguagem Java avaliaram um total de 8 exemplos de trechos de código de testes. Para cada exemplo de trecho de código, era proposto aos participantes que fizessem a análise do exemplo buscando nele algum problema, e depois eram instigados a dar mais detalhes sobre o que enxergaram como inconsistência. Após as apresentações dos problemas e as discussões propostas sobre eles serem finalizadas, foi introduzido e explicado o conceito de *Test Smells*, a qual tipo os exemplos se encaixavam, e como caracterizá-los. Tendo em vista que foi utilizada a técnica de grupo focal, então cada participante tinha o tempo que lhe fosse propício para realizar a análise do problema, e responder o aos questionamentos feitos pelo facilitador. Já toda a entrevista teve uma duração de aproximadamente uma hora.

4.2.1. Análise dos exemplos

Por meio da entrevista, o interesse era entender se desenvolvedores de *software* eram capazes de analisar, reconhecer, e até mesmo resolver os *Test Smells* apresentados. Portanto, em seguida, serão apresentadas as declarações dos participantes sobre os exemplos apresentados para cada tipo de *Test Smells*.

Assertion Roulette. Foi verificado que para o primeiro exemplo 50% dos participantes não enxergaram problema algum, enquanto os outros 50% identificaram no teste muitas asserções, e que o teste poderia estar testando além do necessário. Já no segundo exemplo 75% dos participantes declararam dificuldades em entender a finalidade do teste, visto que a quantidade de asserções estava além do necessário para um código de teste, enquanto 25% dos participantes não identificaram problema algum. Como proposta de correção dos dois exemplos, por parte dos participantes que encontraram problemas, está a refatoração do método de teste, separando as asserções em métodos de testes diferentes, e com uma assinatura mais descritiva, facilitando assim o entendimento do propósito do teste.

Eager Test. Foi verificado que para o primeiro exemplo, 100% dos participantes encontraram problemas como: o método de teste apresentava variáveis com nomes que não faziam jus ao conteúdo atribuído, e o método também estaria testando mais de um método do objeto em questão. Agora, no segundo exemplo 50% dos participantes não encontraram problema, enquanto os outros 50% expuseram que novamente o método de teste estava testando mais de uma unidade do objeto, igual ao primeiro exemplo. Esses participantes também alegaram que não acharam interessante a instância ser global, pois alguns testes poderiam afetar o comportamento desse objeto antes da execução de outro. Para o primeiro exemplo, os participantes propuseram que cada método do objeto fosse testado em um método de testes específico pra si, tornando mais separadas as responsabilidades, bem como a refatoração do nome das variáveis de modo que ficassem de acordo com o valor que lhes fosse atribuído. Para o segundo exemplo, os participantes que enxergaram os problemas descritos acima, também propuseram que cada método do objeto fosse testado em métodos específicos, e que a instância global do objeto fosse transformada em uma

instância local, pois assim diminuiria a chance de que outros métodos afetassem o comportamento de algum teste.

Magic Number. De acordo com 100% dos participantes, o primeiro exemplo não apresentava problema algum em sua implementação. Todos julgaram o teste como um bom teste, de fácil leitura e entendimento. No segundo exemplo, de acordo com 50% dos participantes, o teste também não continha problema algum em sua implementação. Enquanto os outros 50% tiveram dificuldade de entender o propósito do teste devido à quantidade de asserções presentes nele, e propuseram como correção do segundo exemplo, ou separar as diversas asserções em métodos de testes diferentes, ou agrupar asserções parecidas dentro do mesmo teste, em busca de facilitar a leitura e o entendimento do propósito do teste.

Lazy Test. De acordo com 100% dos participantes, não existia problema algum com o primeiro exemplo. Já para o segundo exemplo 25% dos participantes também não enxergaram problema no teste. Porém, os outros 75% dos participantes informaram achar a nomenclatura do método muito genérica, ou seja, pouco específica sobre qual de fato seria a ação a ser executada por ele, tornando mais complexa sua a leitura e seu entendimento. Como proposta de solução para o problema encontrado, os participantes sugeriram que o nome do método de teste fosse alterado para um nome mais descritivo, e que de fato indicasse qual seria a sua finalidade.

4.2.2. Análise dos questionamentos

Durante a entrevista, juntamente com o grupo focal, e após as discussões sobre os exemplos de *Test Smells* apresentados, a intenção era entender se desenvolvedores que atuam na indústria tinham consciência do termo *Test Smells*, se eles consideravam esse tema relevante no seu dia-a-dia, se enxergavam *Test Smells* como um potencial problema para a suíte de testes de um projeto, bem como se seriam a favor de ferramentas que auxiliassem a identificação e até mesmo a correção de ocorrências de *Test Smells*.

Consciência do termo *Test Smells* (Q1). No grupo focal, foi investigado o conhecimento dos participantes sobre o termo *Test Smells*. Durante a entrevista os participantes foram questionados se eles conheciam ou se tinha consciência do que vinha a ser *Test Smells*. Portanto, foi observado que a maioria dos participantes (75%) não tinham conhecimento sobre o conceito de *Test Smells*. O restante dos participantes (25%) alegou conhecer o termo *Test Smells*, indicando que seriam pontos que dificultam a legibilidade e o entendimento dos códigos de teste e fizeram uma analogia com o termo *Code Smells*.

Relevância de *Test Smells* em projetos (Q2). Na entrevista, buscou-se entender se para os participantes, *Test Smells* teriam alguma relevância nos projetos em que trabalham ou desenvolvem no dia-a-dia. Essa discussão nos mostrou que todos os participantes (100%) consideraram o tema relevante, e que deveria ser levado em conta durante o desenvolvimento de uma súite de testes de um projeto. Como justificativas, boa parte dos participantes (66,7%) comentou buscar fontes para aprender sobre o tema e, assim, poder aplicar esse conhecimento em seus projetos. Já (33,3%) dos participantes alegaram que *Test Smells* servem como parâmetro para pautar re-

visões, pois é interessante de se trabalhar baseado em uma métrica ou teoria que seja estruturada.

Interesse na resolução dos exemplos apresentados (Q3). Todos os participantes (100%) afirmaram ter interessem em resolver os exemplos de *Test Smells* apresentados na primeira etapa da entrevista. Os participantes alegaram que a correção de cada um dos exemplos tornaria mais fácil o entendimento, legibilidade e manutenção dos testes. Uma pequena parte dos participantes (33,3%) comentou também que os testes funcionam como uma documentação viva de um projeto, e com eles seria possível enxergar o propósito de uma unidade, possibilitando uma visão mais ampla do projeto.

Opinião sobre ferramentas auxiliares (Q4). Conforme observado, todos os participantes (100%) consideraram válido o uso de ferramentas que os auxiliasse na identificação, e até mesma na correção das ocorrências de *Test Smells*. Todos os participantes comentaram que o uso dessas ferramentas nos projetos em que atuam, traria um ganho considerável nas atividades de refatoração dos métodos de teste, assim, poupando muito tempo, pois a ferramenta automatizaria um processo que hoje é feito de forma manual. Os participantes também reforçaram que essas ferramentas serviriam como uma referência, trazendo uma linguagem em comum aos seus usuários, servindo como uma base pra medir e executar as análises nos códigos de teste de uma aplicação.

5. Ameaças à Validade

Validade interna. Embora a quantidade de *Test Smells* não seja pequena, esse estudo levou em consideração somente 4 deles. Entretanto, os que foram selecionados tem sido frequentemente discutidos na literatura. Para refinar o roteiro do estudo e diminuir os equívocos sobre os questionamentos, foi aplicado um piloto com um desenvolvedor da mesma companhia e atuante no mesmo projeto.

Validade externa. O estudo foi realizado com 4 profissionais e um único *software*. Embora não seja possível usar esses resultados de maneira geral, eles proporcionam uma visão inicial sobre as percepções de desenvolvedores de software que atuam na indústria sobre *Test Smells*. Tendo em vista as limitações, os procedimentos do estudo foram executados e os dados disponibilizados para permitir novas adaptações e replicações do estudo.

6. Conclusão

Nesse artigo são apresentados os resultados de um estudo de método misto, ou seja, utilizando métodos quantitativos e qualitativos, sobre a visão de desenvolvedores de *software* que atuam na indústria em relação ao tema *Test Smells*.

No primeiro estudo, foram coletados dados relativos a um projeto desenvolvido inteiramente em linguagem Java pela indústria. E com isso, foi possível verificar que um grande projeto da indústria, com muitas linhas de código contém uma quantidade considerável de *Test Smells* implementados em suas súites de testes unitários. Nesse caso, aproximadamente 88,24% das classes de testes carregavam em si pelo menos um tipo de *Test Smells*.

No segundo estudo, foi entrevistado um grupo de desenvolvedores que atuam ou atuaram no projeto utilizado como base, de acordo com a abordagem de grupo focal, que primeiramente analisaram 8 exemplos de *Test Smells*. As suas respostas e justificativas nos levam a concluir que mesmo sem haver conhecimento sobre o termo *Test Smells*, desenvolvedores de *software* com consideráveis níveis de experiência são capazes de reconhecer, e até mesmo apresentar soluções para trechos de código mal desenvolvidos em códigos de teste. Da mesma forma, esse estudo indica que nem todos os tipos de *Test Smells* são considerados um problema para os desenvolvedores, ou seja, eles conseguem ler e entender o propósito de um código de teste que represente algum tipo de *Test Smells*. As descobertas também indicam que desenvolvedores de *software* que atuam na indústria consideram *Test Smells* um tema relavante e a ser levado em conta nos projetos que atuam diariamente, bem como as resoluções desses problemas, principalmente por meio de ferramentas que automatizem esse processo de identificação e correção de *Test Smells*, trazendo mais qualidade aos códigos de testes, de uma forma extremamente mais ágil e produtiva.

Como trabalho futuro, a prentensão é aumentar a lista de projetos que sirvam de base para a coleta de exemplos de *Test Smells*, bem como o número de desenvolvedores de *software* que possam expor suas percepções sobre o tema, avaliar a importância do conceito nos projetos em que atuam, e por fim, apoiar na investigação sobre a relevância de ferramentas que auxiliem na busca e correção de *Test Smells*.

Declarações

Grande parte desse trabalho foi conduzido pelo primeiro autor desse artigo (Gabriel Crespo de Souza), durante o seu projeto final como graduando em Ciência da Computação (Bacharelado), na Universidade de Brasília (UnB).

Referências

Bavota, G., Qusef, A., Oliveto, R., Lucia, A. D., and Binkley, D. W. (2012a). An empirical analysis of the distribution of unit test smells and their impact on software maintenance. In *28th IEEE International Conference on Software Maintenance, ICSM 2012, Trento, Italy, September 23-28, 2012*, pages 56–65. IEEE Computer Society.

Bavota, G., Qusef, A., Oliveto, R., Lucia, A. D., and Binkley, D. W. (2012b). An empirical analysis of the distribution of unit test smells and their impact on software maintenance. In *28th IEEE International Conference on Software Maintenance, ICSM 2012, Trento, Italy, September 23-28, 2012*, pages 56–65. IEEE Computer Society.

Bavota, G., Qusef, A., Oliveto, R., Lucia, A. D., and Binkley, D. W. (2015). Are test smells really harmful? an empirical study. *Empir. Softw. Eng.*, 20(4):1052–1094.

Campos, D., Soares, L. R., and Machado, I. (2021). Developers perception on the severity of test smells: an empirical study. *CoRR*, abs/2107.13902.

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

JetBrains (2001). Intellij idea homepage. `https://www.jetbrains.com/idea/`. Accessed: 2023-04-04.

Junior, N. S., Martins, L. A., Rocha, L., Costa, H. A. X., and Machado, I. (2021). How are test smells treated in the wild? A tale of two empirical studies. *J. Softw. Eng. Res. Dev.*, 9:9:1–9:16.

Junior, N. S., Soares, L. R., Martins, L. A., and Machado, I. (2020). A survey on test practitioners' awareness of test smells. *CoRR*, abs/2003.05613.

Kent Beck, E. G. (1997). Junit homepage. `https://junit.org/junit5/`. Accessed: 2023-04-04.

Kontio, J., Lehtola, L., and Bragge, J. (2004). Using the focus group method in software engineering: Obtaining practitioner and user experiences. In *2004 International Symposium on Empirical Software Engineering (ISESE 2004), 19-20 August 2004, Redondo Beach, CA, USA*, pages 271–280. IEEE Computer Society.

Palomba, F., Nucci, D. D., Panichella, A., Oliveto, R., and Lucia, A. D. (2016). 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, SBST@ICSE 2016, Austin, Texas, USA, May 14-22, 2016*, pages 5–14. ACM.

Peruma, A. S. A. (2018). *What the smell? an empirical investigation on the distribution and severity of test smells in open source android applications*. Rochester Institute of Technology.

Santana, R., Fernandes, D., Campos, D., Soares, L., Maciel, R. S. P., and Machado, I. (2021). Understanding practitioners' strategies to handle test smells: a multimethod study. In Vasconcellos, C. D., Roggia, K. G., Collere, V., and Bousfield, P., editors, *35th Brazilian Symposium on Software Engineering, SBES 2021, Joinville, Santa Catarina, Brazil, 27 September 2021 - 1 October 2021*, pages 49–53. ACM.

Topinka, T. (2009). Statistic page. `https://plugins.jetbrains.com/plugin/4509-statistic`. Accessed: 2023-04-04.

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.

van Zyl, J. (2004). Maven homepage. `https://maven.apache.org/`. Accessed: 2023-04-04.