Código-fonte ou Casos de Teste: onde e a quem devem?

Diogo Alves de Moura Loiola[1], **Alcemir Rodrigues Santos**[1]

[1]Universidade Estadual do Piauí (UESPI) – Campus Piripiri

diogoloiola@prp.uespi.br, alcemir@prp.uespi.br

Resumo. *Dívida Técnica (DT) se referem às tarefas por fazer durante desenvolvimento e evolução de testes, a exemplo dos TODOs esquecidos pelo código. É possível identificar esta DT através da mineração de comentários. No entanto, as soluções identificadas consideram o repositório como um único artefato e desconsideram as diferenças entre os artefatos de código-fonte e casos de teste. Neste artigo resolveu-se explorar esta diferença. Utilizou-se a ferramenta* EX-COMMENT *para minerar dívidas em repositórios de código aberto. Após as analises, concluiu-se que existe diferença na frequência dos tipos de DT, mas há um padrão quanto à distribuição dos tipos de DT.*

1. Introdução

O processo de desenvolvimento de software pode estar propenso a falhas, essas falhas podem estar ligadas a fatores como, tempo curto de entrega, documentação incompleta/insuficiente e estouro de orçamento. Infelizmente as dificuldades que acontecem nesse processo fazem com que os desenvolvedores usem de práticas impróprias em favor da conclusão de suas atividade, no primeiro instante irá se ter uma ganho de tempo, mas à medida que o desenvolvimento avança e esses problemas não são resolvidos, o software se tornará difícil de manter e evoluir no futuro. A comunidade de engenharia de software chamam estes problemas com os artefatos de software de Dívida Técnica (DT) [Spínola et al. 2013].

As consequências de instâncias de DT podem afetar o valor de um software, os custos de manutenção futura, o planejamento e a qualidade do software [Avgeriou et al. 2016]. Esta observação, somada ao fato de que a DT pode acontecer em qualquer projeto, e que apenas certos desenvolvedores sabem onde se encontram esses problemas, fazem com que um projeto de software mal planejado seja fadado ao fracasso.

Por estes motivos, diversos trabalhos vem atacando este problema [de O. Passos et al. 2018, Mendes et al. 2019, Maldonado and Shihab 2015]. Uma iniciativa recente [de O. Passos et al. 2018] buscou identificar DT automaticamente em comentários de código-fonte através de mineração de dados. Com objetivo identificar DT no documento de requisitos do sistema, utilizaram abordagem manual e automatizada no estudo, chegaram a conclusão que é possível identificar DT nos documentos de requisitos. O estudo realizado por mendes e seus colegas em [Mendes et al. 2019] propuseram uma criação de uma ferramenta que os auxiliasse no processo de identificação e evolução nos itens de DT. Em todos os estudos mostrados de mineração, eles utilizam os repositórios como um único artefato de software. Mas dentro de um repositório podemos dividi-lo em dois, que são os artefatos de código-fonte e artefatos de casos de teste. Estudo proposto por Maldonado e seus colegas [Maldonado and Shihab 2015] fizeram um analise de comentários em 5 repositórios de código-fonte aberto, ao todo foram mais 166 mil

comentários após um filtro nos comentários restou 33 mil, com esse estudo foi possível identificar 5 tipos de DT.

Com o objetivo de explorar esses dois artefatos, nesse estudo experimental envolveu-se a ferramenta EXCOMMENT de código-aberto para minerar DT em um *dataset* coletado através de ferramenta própria. Com o objetivo de explorar as diferenças entre as DT identificadas nos dois tipos de artefatos isoladamente.

Os resultados desse estudo indicam que existem diferenças na frequência de DT entre os artefatos, e também há um padrão nas distribuições de DT, mesmo os repositórios possuindo características diferentes. As contribuições deste trabalho, podem ser enumeradas da seguinte maneira:

Dataset: a construção de pequeno *dataset* para análises futuras;
Distinção da natureza dos artefatos: ao apontar as diferenças na concentração de tipos de DT em código-fonte e casos de teste, colabora-se na direção da melhor gestão dos tipos específicos de dívidas durante o ciclo de vida do software.
Direções para pesquisa futura: este é um estudo exploratório, e portanto, um dos resultados é iluminar novas direções de pesquisa;

O restante do artigo está assim organizado. A Seção 2 apresenta conceitos importantes para compreensão do estudo. A Seção 3 apresenta os trabalhos julgados como relacionados a este. Em seguida, a Seção 4 apresenta o planejamento e a execução do estudo experimental de replicação. A Seção 5 apresenta os resultados alcançados com o estudo, enquanto a Seção 6 discute os mesmos à luz da interpretação dos autores. Por fim, a Seção 7 apresenta as ameaças à validade do estudo e a Seção 8 conclui este artigo e aponta direções para trabalhos futuros.

2. Fundamentação teórica

Nesta Seção apresenta-se os conceitos centrais deste trabalho, de maneira a facilitar o entendimento do seu conteúdo. Inicia-se com a conceito de *Dívida Técnica*. Em seguida, apresenta-se a *Mineração de repositórios* e por fim o conceito de *Teste de Software* .

2.1. Dívida Técnica

Uma *(DT)* é problema que o desenvolvedor deixa para fazer depois para ganhar algum tempo no desenvolvimento. Sabendo que irá ter um custo a ser pago por ter deixado essa atividade para ser feita depois. É possível fazer um paralelo da DT com um empréstimo financeiro, com algumas diferenças. No empréstimo financeiro obtém-se uma determinada quantia de dinheiro, conhecendo o valor a ser pago e o tempo para quitar tal divida.Na DT não ocorre o mesmo, em um projeto de software, não se sabe quando a DT será paga, e nem o custo que ela trará ao time de desenvolvimento. [Guo and Seaman 2011]. Alves e seus colegas [Alves et al. 2016] falam que DT não se apresenta apenas em código-fonte que não segue boas recomendações, ela pode ocorrer de diferentes maneiras como, casos de teste, documentação incompleta/insuficiente, e arquitetura mal planejada.

2.2. Mineração de repositórios

Repositórios de software armazenam artefatos de software que foram produzidos durante desenvolvimento e evolução do software. Existem diferentes tipos de repositório: (*controle de origem*, *bugs*, *logs de implantação* e *código*). Somente o último é do interesse

deste estudo experimental, uma vez que, nestes, têm-se acesso aos comentários deixados no artefatos de código-fonte e de casos de teste. A *Mineração de repositórios* se refere ao estudo exploratório destes artefatos com o propósito de a entender melhor o processo de software [Kagdi et al. 2007].

2.3. Código-fonte *versus* Casos de Teste

O teste de software é uma atividade que tem como o objetivo verificar se ele está de acordo com as especificações que foram planejadas, e se funciona corretamente no ambiente em que será implantado. O objetivo do teste e revelar *bugs*/falhas no sistema.[NETO 2007]. O principal artefato gerado na atividade de teste é o caso de teste. Eles são métodos que verificam uma pequenas partes da especificação de requisitos de forma independente.

É possível entender os casos de teste como parte integrante do código-fonte de um software, mas os propósitos distintos destes artefatos permitem que os tratemos de maneira distinta. Especialmente, pelo fato de que cada um tem um impacto diferente no processo de software. Esta é a razão pela qual faz-se esta distinção na interpretação da dívida técnica destes artefatos nesse estudo.

3. Trabalhos Relacionados

Nesta Seção apresta-se os trabalhos relacionados a este. De maneira alguma, pretende-se fazer uma lista exaustiva, mas sim apresentar os que tiveram influência na construção deste. Passos e seus colegas [de O. Passos et al. 2018], que utilizaram a ferramenta EX-COMMENT para descobrir padrões e identificar DT nos documentos de requisitos através de análise qualitativa e quantitativa dos padrões identificados.

Mendes e seus colegas apresentam o VISMINERTD, uma ferramenta que permite a automação de identificação e monitoramento interativo da evolução dos itens de DT. Eles avaliaram a ferramenta através de um estudo de caso onde analisaram repositórios de código com a ferramenta. Os resultados indicam que VISMINERTD pode auxiliar no acompanhamento de itens de DTs e evolução de software, a ferramenta também se provou eficiente já que para grandes projetos de software ela concluiu a tarefa em minutos. [Mendes et al. 2019]

O trabalho realizado por maldonado e seus colegas [Maldonado and Shihab 2015], fez a identificação de itens de DT auto-admitida, para realizar esse estudo foi analisado 5 repositórios de código-fonte aberto que tinham diferentes domínios, ao todo foram mais de 166 mil comentários analisado, depois de uma filtragem sobraram 33 mil comentário., Foram identificado ao todo um conjunto de cinco tipos de DT, DT de projeto, DT de documentação, DT de teste, DT de exigência, DT de defeito.

Os estudos citados, embora tenham utilizado comentários de código para a identificação de DT, não contemplaram uma análise detalhada sobre a diferença entre as DT identificadas no código fonte e aquelas dos casos de teste. Nosso estudo contemplou as diferenças existentes nas DT identificadas nos artefato de código-fonte e nos casos de teste.

4. Estudo Experimental

Este estudo experimental tem como objetivo explorar a diferença, se existir alguma, entre as DT, que podem ser identificadas nos comentários deixados no código-fonte e naqueles

deixados nos casos de teste no contexto de sistemas de código aberto e com relação à frequência e a distribuição em que ocorrem. Neste sentido, estabeleceu-se as seguintes questões de pesquisa:

QP1 Existe diferença quanto à frequência de DT entre os artefatos de código-fonte e de casos de teste?

QP2 Existe diferença quanto à distribuição de DT entre os artefatos de código-fonte e de casos de teste entre sistemas?

Para atacar **QP1** e **QP2**, planejou-se este estudo tomando como referência os *parâmetros* de Wohlin e seus colegas [Wohlin et al. 2012]. A Figura 1 apresenta as etapas conduzidas na realização do estudo. Primeiro, fez-se a coleta dos repositórios do *GitHub*, para a criação do *dataset* a ser analisados. Em seguida, fez-se a separação dos artefatos código-fonte e dos casos de teste. Na terceira etapa, utilizou-se a ferramenta EXCOMMENT [de O. Passos et al. 2018] para análise dos comentários em busca de padrões. Na quarta etapa, fez-se a consolidação dos dados produzido pela EXCOMMENT . Por fim, apresenta-se os resultados.

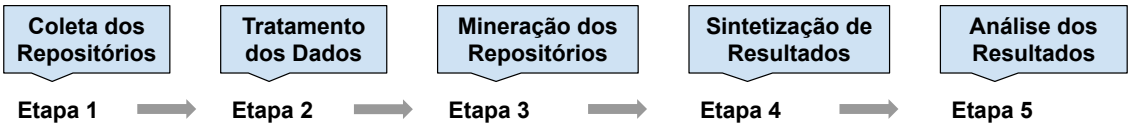

Figura 1. Etapas executadas no estudo experimental.

4.1. Coleta de Repositórios e Tratamento dos Dados

Para a coleta dos repositórios, implementou-se a GIFT , que permite buscar os repositórios no *GitHub* e facilita a criação de um *dataset*, para um determinado propósito, através da busca avançada utilizado os parâmetros disponibilizados na API do *GitHub*. Com a ajuda da mesma, utilizou-se os seguintes parâmetros para selecionar os repositórios: *(i)* data da criação à partir de 2011; e *(ii)* linguagem de programação predominante `Java`, pois a EXCOMMENT minera apenas repositórios com a linguagem java.

Após coleta dos repositórios, fez-se a separação dos artefatos de código-fonte e de casos de teste. Outros passos, de tratamento dos dados coletados, são automatizados pela EXCOMMENT , como a remoção de acentos, caracteres especiais, radicais e palavras de ligação do conteúdo a ser analisado, os comentários.

4.2. Sistemas Alvo

A Tabela 1 apresenta uma caracterização dos versão dos sistemas utilizados neste estudo. A coluna *UC* indica o último `commit`. a *CT* indica total de comentários extraídos. Deste total, a coluna *CV* aponta os comentários válidos, aqueles que foram efetivamente úteis na avaliação, enquanto a coluna *LOC* aponta o tamanho dos sistemas em quantidade de linhas de código. Os dados apresentados na coluna indicativa da quantidade de *CV* foi preponderante para a seleção destes sistemas como alvo. Para seleção os repositórios foram aplicados alguns critérios, ter mais de 3 anos de duração e ter uma quantidade de comentários superior a 10.000 comentários, e pertencer a diferentes domínios de aplicação e variar em tamanho.

Tabela 1. Caracterização dos sistemas-alvo.

Repositório	Domínio	Versão	A	UC	CT	CV	LOC
TOMCAT	*SW*	7.0.99	CF	a94a025	35316	23598	233040
			CT	a94a025	19472	11104	60830
HBASE	*BD*	2.2.3	CF	6a830d8	41997	31807	429764
			CT	6a830d8	43494	27052	303363
HIVE	*DW*	3.1.3-rc0	CF	66aeb5b	74591	49569	1017585
			CT	66aeb5b	14205	8232	236294
PRESTO	*CSQL*	0.233.1	CF	2d75403	32930	19920	490518
			CT	2d75403	40821	24022	235098
NIFI	*DD*	1.11.1	CF	d22858d	68561	43253	418507
			CT	d22858d	78078	48813	199677

SW: Servidor Web; **BD:** Banco de Dados; **DW:** Data Warehouse; **CSQL:** Consultas SQL em BigData ; **DD:** Processador e distribuidor de dados; **A:** Artefato; **CF:** Código-fonte; **CT:** Casos de teste; **UC:** Último commit; **CT:** Total de comentários; **CV:** Comentários válidos; **LOC:** Tamanho em linhas de código.

4.3. Mineração de Dívida Técnica em Repositórios de Software

A mineração de DT foi realizada com a EXCOMMENT , que utiliza técnicas de mineração de texto e processamento de linguagem natural. O funcionamento da ferramenta é dividido em 3 partes, *(i)* pré-processamento; *(ii)* estratégia de busca, e *(iii)* classificação de itens de DT. A etapa de pré-processamento, ela irá separar os comentários úteis para a mineração, e descarta o restante. Depois ela irá fazer uma associação entre os comentários úteis a trechos do código fonte. A segunda etapa ela vai fazer uma busca automática de itens de DT, utilizando de métricas pré-estabelecidas que a EXCOMMENT disponibiliza. Na última etapa, cada possível ocorrência de DT identificado na etapa anterior é classificado pela ferramenta [de O. Passos et al. 2018]. Os tipos de DT considerados neste estudo foram: *(DT3)* problemas relacionados a qualidade do código; qualidade; Débito *(DT4)* problemas identificados através de atividades automatizadas; design *(DT5)* problemas relacionados ao desacoplamento; documentação *(DT6)* problemas como documentação inadequada/incompleta; requisitos *(DT8)* Saber o que se deve mudar e como implementar; e teste *(DT9)* podem indicar uma baixa cobertura ou testes que não estão sendo executados[Mendes et al. 2019].

5. Resultados

Nesta Seção, apresenta-se os resultados da mineração de DT nos repositórios. A Figura 2 apresenta a frequência dos diferentes tipos de DT encontradas nos sistemas-alvo analisados. A literatura que foi utilizada para classificar os tipos de DT possuem 9 tipos [Mendes et al. 2019], os tipos DT1, DT2 e DT7 foram omitidos pois quantidade de DT identificada foi nula ou muito pequena com relação aos demais tipos. Os artefatos de código-fonte tiveram uma quantidade maior de DT do que os artefatos de casos de teste. Esta diferença era esperada, visto que mais comentários no código-fonte do que nos casos de teste (Tabela 1). Quanto aos artefatos de casos de teste, percebemos que a maior concentração de DT9, que trata de DT relacionada à atividade de teste.

A Tabela 2 mostra a distribuição da porcentagem de cada tipo de DT nos sistemas-alvo analisados. Os tipos DT1 e DT7 foram omitidos pois quantidade de DT identificada

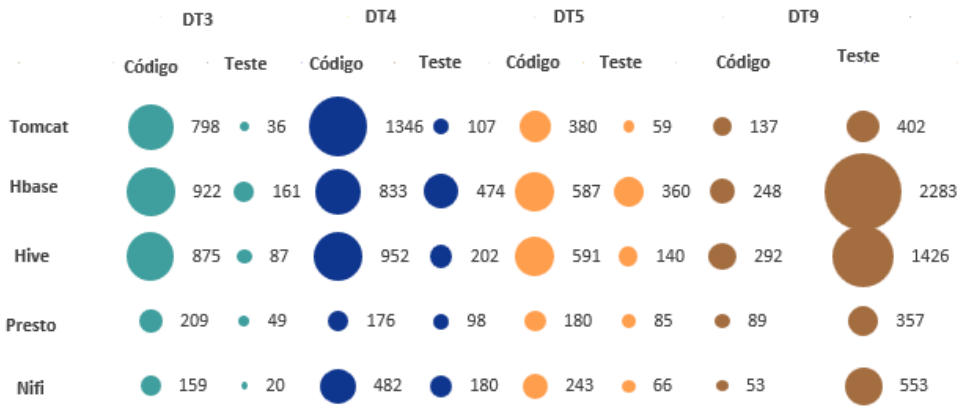

Figura 2. Frequência de DT nos artefatos de código-fonte e casos de teste.

foi nula ou muito pequena com relação aos demais tipos. Mais de 50% das ocorrências de DT identificada são do tipo DT3 e DT4 em todos os sistemas-alvo. O tipo DT5 também representa quantidade significante de DT em todos os sistemas-alvo. Por fim, identificou-se pouca ocorrência dos tipos DT6 e DT8 nos sistemas-alvo. Todas as porcentagens de distribuição aparecem independente do tipo de artefato analisado.

Tabela 2. Distribui dos tipos de DT (em %).

Repositório	Artefato	DT3	DT4	DT5	DT6	DT8	DT9	Total*
TOMCAT	CF	29.8	50.2	14.2	0.3	0.3	5.1	100% (2680)
	CT	5.9	17.7	9.7	0	0.3	66.3	100% (607)
HBASE	CF	35.0	31.6	22.3	0.2	1.3	9.4	100% (2635)
	CT	4.9	14.4	10.9	0.2	0.1	69.4	100% (3289)
HIVE	CF	31.5	34.3	21.3	0.9	1.4	10.5	100% (2778)
	CT	4.7	10.9	7.5	0.1	0	76.8	100% (1858)
PRESTO	CF	31.5	26.5	27.1	0.8	0.6	13.4	100% (664)
	CT	8.3	16.6	14.4	0.5	0	60.3	100% (593)
NIFI	CF	16.6	50.2	25.3	1.9	0.5	5.5	100% (961)
	CT	2.4	21.8	8.0	0.5	0.1	67.1	100% (825)

A: Tipo de artefato; **CF:** Código-fonte; **CT:** Casos de Teste; **DT2:** DT de Construção; **DT3:** DT de Código; **DT4:** DT de Defeito; **DT5:** DT de Design; **DT6:** DT de Documentação; **DT8:** DT de Requisitos; **DT9:** DT de Teste; ***:** Número absoluto de dívida técnica no tipo de artefato do sistema alvo.

6. Discussão

Dados os resultados apresentados, faz-se agora uma discussão dos mesmos, de maneira a responder as questões de pesquisa deste trabalho.

QP1: Existe diferença quanto à frequência de DT entre os artefatos de código-fonte e de casos de teste?

De acordo com a Figura 2, *existe diferença quanto à frequência de DT nos artefatos de código-fonte e casos de teste*. Nos artefatos de código-fonte, os tipos DT3, DT4 e DT5 possuem maior ocorrência em relação aos artefatos de casos de testes. Enquanto,

o tipo DT9 aparece com maior frequência nos artefatos de casos de teste. Acredita-se que isto aconteça por que no código-fonte encontra-se toda a lógica da aplicação e nos artefatos de casos de teste o objetivo central é verificar se a implementação está de acordo com a especificação. Além disso, os comentários tendem a ser mais extensos e detalhados nos artefatos de código-fonte.

Este elevado número de DT nos artefatos de código-fonte e de casos de teste dos sistemas analisados mostra que uso de más práticas de codificação aparentam ser recorrentes. Por exemplo, *(i)* problemas relacionados *design* do sistema (DT5) não respeitando o desacoplamento e classes pequenas; ou ainda *(ii)* aqueles relacionados com os casos de teste (DT9), podem indicar baixa cobertura ou testes não executados. Seria necessário confirmá-las com com os desenvolvedores. Estas dívidas podem afetar diretamente o funcionamento do sistema.

QP2: Existe diferença quanto à distribuição de DT entre os artefatos de código-fonte e de casos de teste entre sistemas?

De acordo com a Figura 2, *não existe diferença quanto à distribuição de DT nos artefatos de código-fonte e casos de teste*. O que se percebe, é a repetição de um padrão nas distribuições das DT nos artefatos. A média da porcentagem dos tipos de DT é similar em todos os repositórios.

Analisando os repositórios TOMCATe HIVEda Tabela 2, que são repositórios que não pertencem aos mesmos domínios de aplicação, possuem tamanhos variados e quantidade de DTs diferentes. Mesmo com todos esses fatores de diferença percebe-se, que esses fatores não influenciaram as distribuições DT. Uma possível justificativa para esse fato é, que independente do tamanho do repositório, domínio e quantidade DT, os repositórios sempre apresentam presença dos mesmos itens de DT.

7. Ameaças à Validade

Embora estudos empíricos tenham que ser exaustivamente planejados para aumentar a confiança do estudo [Wohlin et al. 2012], nem sempre é possível contornar todas as variáveis existentes. Este estudo não foge à realidade. Desta forma, faz-se necessário a discussão de algumas ameaças à validade que podem ser identificadas neste estudo. São elas: *(i)* para estudos de mineração de dados em repositório de software, o ideal seria executar o estudo com a maior base de dados possível, no entanto, apesar de termos identificado quantidade razoável de casos de teste automatizados com o uso da GIFT , a quantidade de comentários destes casos de teste pode ter dificultado a identificação de mais dívidas técnicas no contexto de testes, o que minimizamos, ao selecionar repositórios com quantidade razoável de comentários; *(ii)* na etapa de construção do *dataset* tivemos dificuldades de encontrar repositórios representativos e que ao mesmo tempo tivessem quantidade equivalente de comentários no artefatos de código-fonte e de casos de teste; a *(iii)* limitação classificação de DT pela EXCOMMENT , seja pelos algoritmos, vocabulários e/ou características escolhidas em sua implementação; e a *(iv)* existência de um único tipo de DT relacionado especificamente aos casos de teste.

8. Conclusão e Trabalhos Futuros

Nesse estudo foi feito um comparativo entre os artefatos de código-fonte e casos de teste através da mineração de comentários. Pode-se afirmar que há uma diferença de frequência

de DT entre os artefatos. E que não há uma diferença nas distribuições de DT entre os artefatos. Observou-se que nos artefatos que possuíam uma quantidade maior de linhas de código também possuíam uma grande quantidade de comentários, tendo implicação nas quantidades de DT encontradas nos repositórios. Mesmo os repositórios possuindo tamanhos variados, e sendo de domínios diferentes, percebeu-se que esse fato não alterou as distribuições de DT. Os resultados alcançados são interessantes e permitem enxergar novas direções de trabalho. Pretende-se ampliar a base de repositórios para reforçar os achados, identificar e se há correlação entre tamanho e a frequência e/ou a distribuição das DT. Além disso, é possível ainda tentar identificar tipos específicos de DT nos artefatos de teste.

Referências

Alves, N. S., Mendes, T. S., de Mendonça, M. G., Spínola, R. O., Shull, F., and Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, 70:100–121.

Avgeriou, P., Kruchten, P., Ozkaya, I., and Seaman, C. (2016). Managing technical debt in software engineering (dagstuhl seminar 16162). In *Dagstuhl Reports*, volume 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

de O. Passos, A. F., de Freitas Farias, M. A., de Mendonça Neto, M. G., and Spínola, R. O. (2018). A study on identification of documentation and requirement technical debt through code comment analysis. In *Proceedings of the 17th Brazilian Symposium on Software Quality*, pages 21–30.

Guo, Y. and Seaman, C. (2011). A portfolio approach to technical debt management. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, pages 31–34.

Kagdi, H., Collard, M. L., and Maletic, J. I. (2007). A survey and taxonomy of approaches for mining software repositories in the context of software evolution. *Journal of software maintenance and evolution: Research and practice*, 19(2):77–131.

Maldonado, E. d. S. and Shihab, E. (2015). Detecting and quantifying different types of self-admitted technical debt. In *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, pages 9–15. IEEE.

Mendes, T. S., Gomes, F. G., Gonçalves, D. P., Mendonça, M. G., Novais, R. L., and Spínola, R. O. (2019). Visminertd: a tool for automatic identification and interactive monitoring of the evolution of technical debt items. *Journal of the Brazilian Computer Society*, 25(1):2.

NETO, A. (2007). Introdução a teste de software. *Engenharia de Software Magazine*, 1:22.

Spínola, R. O., Vetrò, A., Zazworka, N., Seaman, C., and Shull, F. (2013). Investigating technical debt folklore: Shedding some light on technical debt opinion. In *2013 4th International Workshop on Managing Technical Debt (MTD)*, pages 1–7. IEEE.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.