Avaliação de Ferramentas de Identificação de Dívida Técnica Auto-Admitida

Tchalisson Brenne S. Gomes
tchalisongomes@aluno.uespi.br
Universidade Estadual do Piauí – UESPI
Piripiri, Piauí, Brasil

Diogo Alves de Moura Loiola
diogoloiola@aluno.uespi.br
Universidade Estadual do Piauí – UESPI
Piripiri, Piauí, Brasil

Alcemir Rodrigues Santos
alcemir@prp.uespi.br
Universidade Estadual do Piauí – UESPI
Piripiri, Piauí, Brasil

RESUMO

Gerenciar dívidas técnicas em um projeto é primordial para saúde de projetos de software. A identificação delas, no entanto, não é trivial. Sejam elas, auto-admitidas ou não, existem diversas ferramentas que se propõem a identificá-las na literatura. No entanto, poucos estudos se propuseram a avaliar a intersecção entre suas abordagens. Neste artigo comparamos os resultados obtidos com duas ferramentas de identificação de dívidas técnicas auto-admitidas que utilizam mineração de comentários de código-fonte: eXcomment e DebtHunter. Utilizamos as ferramentas em quatro sistemas de código-aberto e comparamos manualmente a intersecção e a distribuição da classificação dos itens de dívida de cada uma das ferramentas. Verificou-se que dos itens classificados por eXcomment, 7% deles DebtHunter também classifica como dívida e que, dos itens que DebtHunter classifica como dívida, 19, 9% deles eXcomment também classifica. Além disso, apesar de divergirem quanto à quantidade de itens de dívida técnica, as ferramentas parecem convergir quanto aos tipos de dívida presentes nos sistemas avaliados.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

1 INTRODUÇÃO

Dívida Técnica (DT) é certamente um dos problemas mais investigados pela comunidade de engenharia de software nos últimos anos [1, 2]. Entretanto, a detecção e o gerenciamento de dívida técnica em projetos de software ainda representam um grande desafio. Uma das formas mais utilizadas para atacar esse problema é o desenvolvimento de ferramentas que possam apoiar nestas tarefas. Dentre elas, destacam-se abordagens baseadas em análise estática de código e mineração de comentários de código-fonte [2]. A maioria delas empregam somente abordagens de identificação e medição [1].

Ao mesmo tempo, Avgeriou *et al.* [2] argumentam que a quantidade de ferramentas disponíveis no mercado, sejam comerciais, sejam protótipos de pesquisa, pode se tornar um problema para desenvolvedores que pretendem selecionar uma delas para uso diário. Além disso, abordagens como pagamento das dívidas, monitoramento e priorização das dívidas ainda são bastante limitadas [2]. Se faz necessário, portanto, a avaliação e a comparação dos resultados destas ferramentas existentes para *(i)* garantir a eficácia destas ferramentas, bem como *(ii)* facilitar o processo de escolha de desenvolvedores e *(iii)* ajudar no processo de evolução das mesmas.

O trabalho de comparação entre os resultados de ferramentas de identificação de DT não é trivial. Exige um conjunto de dados robusto e de preferência, sem ambiguidades na classificação dos itens de dívida classificados. Além disso, para cada abordagem de identificação automática de dívida, pode ser necessário um conjunto de dados diferente. Avgeriou *et al.* [2] lançou mão de uma avaliação qualitativa através de discussão em grupo, utilizando-se de três critérios principais: características, popularidade e validação empírica. O resultado mostra que somente SonarQube, DV8 e SonarGraph foram consideradas em estudos empíricos com relação ao índice de DT e que mais estudos são necessários para que se consiga quantificar o esforço para pagar tais dívidas. Em outra iniciativa, Oliveira *et al.* [7] avaliaram duas ferramentas: SonarQube (que utiliza análise estática de código) e SATDDetector (que utiliza a mineração de comentários de código-fonte). O objetivo do estudo foi identificar a intersecção entre ambas as abordagens. Verificou-se que somente $\approx 19,47\%$ dos itens de dívida foram identificadas por ambas as ferramentas avaliadas.

Diferentemente dos trabalhos de Avgeriou *et al.* [2] e de Oliveira *et al.* [7], este artigo apresenta uma comparação entre duas abordagens de identificação de dívida técnica auto-admitida (eXcomment e DebtHunter) que utilizam a mesma técnica, a mineração de comentários de código-fonte. O resultado do comparativo entre a intersecção de abordagens das ferramentas, mostra que dos itens que eXcomment classifica como dívida, em média, somente 7% deles DebtHunter também classifica como dívida. Da mesma forma, dos itens que DebtHunter classifica como dívida, 19, 9% deles eXcomment também classifica.

O restante deste artigo é organizado da seguinte maneira. A Seção 2 apresenta os trabalhos considerados relacionados a este. A Seção 3 descreve como o estudo foi conduzido. A Seção 4 apresenta e discute os resultados encontrados. A Seção 5 apresenta ameaças à validade. Por fim, a Seção 6 apresenta as considerações finais e trabalhos futuros.

2 TRABALHOS RELACIONADOS

Esta seção apresenta os trabalhos que considerou-se relacionados a este [3–5, 7, 10]. Em relação à dívida técnica auto-admitida Bavota e Russo [3] replicaram o trabalho de Potdar e Shihab [8], com o objetivo de identificar a propagação e evolução da dívida técnica auto-admitida. Para isso, foram coletados mais de 600 mil *commits* e dois bilhões de comentários. Os resultados do estudo mostram que a dívida técnica auto-admitida é difusa, em média 51 instâncias por sistema. Verificou-se ainda que é composta, principalmente, por dívida de código e requisito, e que as dívidas aumentam ao longo do tempo devido à introdução de novas instâncias, além de sobreviver por muito tempo no sistema mesmo após serem corrigidas.

Utilizando-se de abordagem automatizada Farias *et al.* [4] realizou um estudo onde utilizou uma ferramenta de identificação de

DT através dos comentários de código-fonte no projeto. O objetivo do estudo era identificar DT de documentação através de comentários. A ferramenta utilizada para o estudo foi a eXcomment uma ferramenta de mineração de código que utiliza técnicas de processamento de linguagem natural. Os resultados concluíram que a ferramenta era capaz de realizar a identificação de DT através da mineração de comentários.

Gomes *et al.* [5] investigaram a relação entre *code smells* e dívida técnica auto-admitida. Para a realização do estudo, utilizou-se três projetos *open-source* desenvolvidos na linguagem Java, ArgoUML, JFreeChart e Apache Ant. Os projetos foram escolhidos devido a sua ampla utilização em estudos acerca de comentários de código-fonte. Como resultado, verificou-se forte tendência entre à dívida técnica auto-admitida e *code smells* e que em alguns casos a utilização de comentários de código-fonte pode complementar informações que não poderiam ser obtidas apenas com o uso de *code smells*.

Em uma iniciativa semelhante, Oliveira *et al.* [7] compararam duas abordagens distintas para identificação de DT. A primeira, uma ferramenta de identificação automatizada de DT, a segunda utilizando-se de comentários deixados pelos próprios desenvolvedores, que admitem a presença de dívida técnica. Os resultados indicam que existe uma certa quantidade de dívidas que podem ser identificadas por ambas abordagens, porém os autores afirmam que ainda faltam estudos para uma maior exatidão do que esses resultados realmente significam.

Zazworka *et al.* [10], compararam a eficácia da detecção de dívida técnica manual, com três ferramentas diferentes. Eles verificaram que as abordagens automatizadas podem ser úteis para identificar certos tipos de dívidas, como *Dívida de Defeito* e *Dívida de Projeto*, mas não são úteis para identificar outros tipos de dívida que foram encontrados pelos desenvolvedores.

Diferente dos trabalhos anteriores, o presente trabalho apresenta uma comparação entre duas abordagens de identificação de dívida técnica auto-admitida (eXcomment e DebtHunter), bem como, apresenta a distribuição dos itens de dívida identificados por ambas as ferramentas.

3 ESTUDO DE CASO

O objetivo deste estudo é **avaliar** *ferramentas de identificação de itens de dívida técnica através da mineração de comentários de código-fonte,* **na perspectiva** *do engenheiro de software,* **com respeito** *à intersecção de itens de dívida encontrados por ambas ferramentas, bem como os tipos mais comuns de dívidas encontradas.* Assim, definiu-se as seguintes questões de pesquisa:

RQ$_1$: Qual a interseção entre as abordagens de identificação automática de dívida técnica auto-admitida das ferramentas eXcomment e DebtHunter?

RQ$_2$: Como são distribuídos os itens de dívida identificados pelas ferramentas eXcomment e DebtHunter?

3.1 Recursos

3.1.1 Ferramentas Avaliadas. Este estudo avalia as ferramentas DebtHunter [9] e eXcomment [4] selecionadas da literatura. Os autores deste estudo não participaram do desenvolvimento de quaisquer das ferramentas avaliadas.

A ferramenta DebtHunter utiliza processamento de linguagem natural e aprendizado de máquina. A classificação é executada em duas etapas: *(1)* Classificação binária, onde são identificados os comentários que possuem ou não dívida técnica auto-admitida e *(2)* classificação multi-classe, onde os comentários que possuem dívida são classificados quanto ao tipo da dívida que expressam. A ferramenta também possui dois casos de uso, são eles: *(1)* Marcação de comentário, neste caso, a classificação é realizada com base no melhor modelo pré-treinado pela DebtHunter e *(2)* Treinamento de novo modelo, neste modo a ferramenta recebe como entrada comentários classificados, que são utilizados para treinar um novo modelo. Para a realização deste estudo de caso, a ferramenta foi utilizada apenas no primeiro caso de uso.

A ferramenta eXcomment utiliza-se de técnicas de mineração de texto para coletar os comentários que possam indicar a presença de dívida técnica. Com essa finalidade, a ferramenta utiliza um vocabulário contextualizado para identificação de dívida técnica em comentários de código-fonte [4]. A ferramenta escolhe os comentários que se relacionem a pelo menos um padrão do vocabulário contextualizado. Por fim, os padrões encontrados em cada comentário são analisados para classificá-los quanto ao tipo de dívida técnica que representam.

3.1.2 Coleta de Sistemas-Alvo. Utilizou-se quatro sistemas-alvo: RocketMQ, Lottie, Trift e Arduino. Os sistemas foram escolhidos em domínios e tamanhos variados, de código-aberto e escritos na linguagem de programação Java. A restrição de linguagem é criada pelas ferramentas avaliadas. A Tabela 1 apresenta uma breve caracterização dos sistemas-alvo selecionados, incluindo *versão*, *domínio*, *tamanho em linhas de código* (LOC), *quantidade de linhas de comentários* (LC), *comentários válidos eXcomment* (CVE) e *comentários válidos DebtHunter* (CVD). Comentários válidos indicam todos os comentários que as ferramentas incluíram em sua busca por itens de dívida técnica durante sua execução.

Tabela 1: Caracterização dos sistemas-alvo.

Sistema (Versão)	**Domínio**	**LOC**	**LC**	**CVE**	**CVD**
Lottie (`v3.5.0`)	Biblioteca Android	49.441	2.704	500	808
RocketMQ (`v4.8.0`)	Mensagens e Streaming	109.261	22.798	1.203	2.215
Trift (`v0.14.1`)	Transporte de dados	311.606	75.494	1.065	1.350
Arduino (`v1.8.16`)	Ambiente de desenvolvimento	149.055	62.643	1.265	2.363

LOC: Quantidade de linhas de código; **LC:** Quantidade de linhas de comentário; **CVE:** Comentários válidos eXcomment; **CVD:** Comentários válidos DebtHunter.

3.2 Execução

O estudo de caso foi executado em duas etapas – *(i)* coleta de dados e *(ii)* síntese dos resultados – detalhadas à seguir.

3.2.1 Coleta de dados. Executou-se as ferramentas eXcomment e DebtHunter nos sistemas-alvos. Cada uma das ferramentas identifica um conjunto diferente de itens de dívida técnica. Enquanto a eXcomment se propõe a identificar nove tipos de dívidas (*Dívida*

de Documentação, *Dívida de Requisitos*, *Dívida de Defeito*, *Dívida de Design*, *Dívida de Teste*, *Dívida da Arquitetura*, *Dívida de Construção*, *Dívida de Código*, *Dívida de Pessoas*) [4], a DebtHunter identifica somente cinco (os cinco primeiros da eXcomment) [9]. Portanto, para este estudo foram considerados apenas os cinco tipos de dívidas encontrados em ambas ferramentas. A Tabela 2 define cada um dos tipos de dívida considerados.

Tabela 2: Definição dos tipos de dívida técnicas auto-admitidas considerados no estudo de caso.

Tipo de Dívida Técnica
Dívida de Documentação: Refere-se a problemas encontrados na documentação do software como ausência de documentação, documentação inadequada ou incompleta.
Dívida de Requisitos: Refere-se a atividades que o time de desenvolvimento tem para implementar e como será realizada a implementação.
Dívida de Defeito: Refere-se a defeitos conhecidos, dos quais o time concorda que deveriam ser corrigidos.
Dívida de Design: Refere-se às más práticas de codificação que violam os princípios da orientação a objetos.
Dívida de Teste: Refere-se a problemas que podem afetar a qualidade das atividades de teste.

Vale ressaltar que, ao analisar os tipos de dívida considerados para identificação pelas ferramentas, percebeu-se uma pequena diferença de nomenclatura. Aquilo que a eXcomment considera *Dívida de Requisitos*, a DebtHunter considera *Dívida de Implementação* da mesma forma que o *dataset* tomado como referência [6] para a construção da mesma. Para efeito de padronização, utilizaremos somente *Dívida de Requisitos* deste ponto em diante, como definido por Rios *et al.* [1] e tomado por referência na eXcomment.

3.2.2 Síntese de resultados. A síntese de resultados consistiu na identificação da intersecção entre as abordagens de identificação de itens de DT. Para isto, foram comparadas as saídas de ambas as ferramentas (*i.e.*, os comentários que de fato as ferramentas apontaram indícios de DT) em duas etapas. Primeiro, tomados os comentários classificados pela eXcomment, comparou-se com a classificação da DebtHunter. Segundo, tomados os comentários classificados pela DebtHunter, comparou-se com a classificação da eXcomment. Esta forma de comparação foi necessária, pois não se identificou uma maneira mais eficiente de comparar as saídas manualmente. Os resultados obtidos após a comparação das saídas das ferramentas serão analisados sob a perspectiva de dois grupos distintos:

- **Grupo 1:** comentários que ambas as ferramentas apontaram DT – este grupo será referenciado como *G1*);
- **Grupo 2:** comentários que somente uma das ferramentas apontou DT – este grupo será referenciado como *G2*).

No *G1* houveram comentários classificados com os mesmos tipos de dívida por ambas as ferramentas (subgrupo $G1_{\text{iguais}}$) e também comentários classificados com divergência entre as ferramentas (subgrupo $G1_{\text{diferentes}}$). Por exemplo, o sistema Lottie contém o seguinte comentário: `/* These flags were in Canvas but they were deprecated and removed. TODO: test removing these on older versions of Android.*/` Neste caso, enquanto a eXcomment apontou que o comentário contém *Dívida de Teste* e *Dívida de Código*, a DebtHunter apontou que contém uma *Dívida de Design*. Ou seja, ambas as ferramentas apontaram que o comentário contém dívida, no entanto, de tipos diferentes. Assim, o comentário seria incluído no grupo ($G1_{\text{diferentes}}$) e não seria incluído no grupo ($G1_{\text{iguais}}$).

No *G2* houveram comentários que foram considerados pela segunda ferramenta livres de dívida técnica ($G2_{\text{livres}}$), bem como comentários que foram considerados inválidos (*i.e.* não foram considerados para classificação – $G2_{\text{inválidos}}$). A Figura 1 mostra um esquema que organiza todos os quatro subgrupos mencionados de *G1* ($G1_{\text{iguais}}$ e $G1_{\text{diferentes}}$) e de *G2* ($G2_{\text{livres}}$ e $G2_{\text{inválidos}}$).

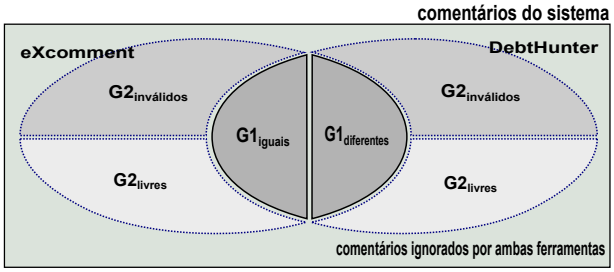

Figura 1: Divisão dos subgrupos utilizados na apresentação e discussão dos resultados.

4 RESULTADOS E DISCUSSÃO

Esta seção apresenta e discute os resultados encontrados durante a comparação entre as ferramentas, bem como os tipos de dívidas encontrados conforme o método definido na Seção 3.

4.1 RQ_1: Interseção entre os resultados das ferramentas eXcomment e DebtHunter

A Tabela 3 apresenta os resultados da comparação entre ambas as ferramentas sob a perspectiva dos dois grupos definidos. A coluna "E" denomina a comparação entre a classificação dos itens de dívida da eXcomment com a classificação da DebtHunter. Analogamente, a coluna "D" denomina a comparação entre a classificação dos itens de dívida da DebtHunter com a classificação da eXcomment.

Como resultados da análise de intersecção entre abordagens, verificou-se que eXcomment detectou o total de 233 itens de dívida técnica, isto somando todos os itens de dívida detectados em todos os sistemas-alvos analisados. DebtHunter, por sua vez, detectou o total de 85 itens de dívida técnica (Tabela 4). Com relação à análise do grupo *G1*, foi possível observar que, em média, a DebtHunter obteve uma intersecção de 7% em relação aos itens de dívida classificados pela eXcomment. Por outro lado, eXcomment obteve uma intersecção de 19, 9% em relação aos itens de dívida classificados por DebtHunter.

Destaca-se o resultado obtido no sistema Trift, onde não foram identificados casos em que ambas as ferramentas classificaram como dívida (*G1*). Na nossa visão, um dos fatores que explicam esse comportamento é o conteúdo dos comentários encontrados

no Trift. Por exemplo, em 14 das 15 *Dívida de Requisitos* encontrados pela DebtHunter foram ao comentário "`TODO(mcslee): implement`". No entanto, como a eXcomment utiliza-se de padrões textuais para a identificação de dívidas [4], os padrões definidos pelas heurísticas da ferramenta não reconhecem itens de dívida em um comentário contendo uma única palavra ("implement").

Quanto a análise do grupo $G2_{\text{livres}}$, dos itens de dívida apontados pela eXcomment, em média, 79, 3% deles a DebtHunter classificou como "livres de dívida técnica". Da mesma forma, dos itens de dívida apontados pela DebtHunter, 27, 1% deles a eXcomment classificou como "livres de dívida técnica". Por fim, quanto ao subgrupo $G2_{\text{inválidos}}$, dos itens de dívida classificados pela eXcomment, 13, 7% deles a DebtHunter considerou inválidos. Por outro lado, dos itens apontados como dívida pela DebtHunter, 63, 6% a eXcomment considerou os mesmos inválidos.

Tabela 3: Análise comparativa da intersecção dos resultados das ferramenta DebtHunter e eXcomment.

	Lottie		RocketMQ		Trift		Arduino	
Grupo	E	D	E	D	E	D	E	D
G1	2	2	2	2	0	0	18	18
➥ $G1_{\text{diferentes}}$	2	2	1	1	0	0	12	12
➥ $G1_{\text{iguais}}$	0	0	1	1	0	0	6	6
G2	36	8	30	15	53	19	92	32
➥ $G2_{\text{livres}}$	31	3	26	0	40	5	87	20
➥ $G2_{\text{inválidos}}$	5	5	4	15	13	15	5	12

E: comparação da classificação dos itens de dívida da eXcomment com a classificação da DebtHunter; **D** comparação da classificação dos itens de dívida da DebtHunter com a classificação da eXcomment.

Após a análise dos itens de dívida classificados pelas ferramentas, verificou-se que eXcomment possui uma porcentagem de intersecção superior em relação aos itens de dívida identificados por DebtHunter. Em média, dos itens de dívida que a eXcomment identificou, DebtHunter conseguiu identificar apenas 7% deles ($G1_{\text{total}}$–E). Enquanto dos itens de dívida identificados pela DebtHunter a ferramenta eXcomment conseguiu identificar 19,9% deles ($G1_{\text{total}}$–D).

De forma geral, percebeu-se nos resultados da avaliação, que a eXcomment tende a ser mais sensível na identificação de itens de dívida que a DebtHunter. Em média, a DebtHunter considerou que não havia dívida ($G2_{\text{livres}}$ – E) em 79, 3% dos comentários contendo dívidas de acordo com a eXcomment. Por outro lado, em média, a eXcomment considerou que não havia dívida ($G2_{\text{livres}}$ – D) em somente 27, 5% dos comentários contendo dívidas de acordo com a DebtHunter.

Percebeu-se também que DebtHunter tende a considerar menos comentários inválidos em relação a eXcomment. Em média, a DebtHunter considerou inválidos ($G2_{\text{inválidos}}$ – E) apenas 13, 7% dos comentários que a eXcomment considerou válidos. Por outro lado, em média, a eXcomment considerou inválidos ($G2_{\text{inválidos}}$ – D) 63, 6% dos comentários considerados válidos segundo a DebtHunter.

Em resumo, a intersecção dos itens de dívida classificados pelas ferramentas eXcomment e DebtHunter é pequena (menor que 20%) em relação a quantidade de itens apontados por cada uma delas.

4.2 RQ_2: Distribuição dos itens de dívida identificados pela ferramentas eXcomment e DebtHunter

A Tabela 4 apresenta o tipo de dívida e a quantidade de itens de dívida técnica identificada por cada uma das ferramentas nos sistemas-alvo. A coluna "tipo de dívida" apresenta qual o tipo de dívida que os itens representam, já a coluna "ferramenta" indica qual a ferramenta essa dívida ocorreu, a *coluna* "TOTAL" indica o total de dívidas por tipo em cada ferramenta. As colunas seguintes indicam em qual sistema os itens de dívida técnica ocorreram. Por fim, a *linha* "TOTAL" indica o total de itens de dívida técnica identificados em cada ferramenta.

Analisando-se a distribuição dos tipos de dívida identificada por ambas as ferramentas, verificou-se poucos ou nenhum item dos tipos *Dívida de Documentação* e *Dívida de Teste*. Enquanto somente a eXcomment encontrou *Dívida de Teste*, nenhuma das ferramentas identificaram itens de *Dívida de Documentação*. Por outro lado, ambas ferramentas encontraram diversos itens de *Dívida de Design*, *Dívida de Defeito* e *Dívida de Requisitos*. Enquanto a eXcomment encontrou mais itens de *Dívida de Defeito*, *Dívida de Design* e *Dívida de Requisitos* (ordem decrescente), a DebtHunter encontrou mais itens de *Dívida de Design*, *Dívida de Requisitos* e *Dívida de Defeito* (ordem decrescente).

Tabela 4: Quantidade de itens de dívida identificados por cada ferramenta em cada sistemas-alvo.

Tipo de Dívida	Ferramenta	Lottie	RocketMQ	Trift	Arduino	TOTAL
Dívida de Design	eXcomment	6	3	6	14	29
	DebtHunter	5	16	4	30	55
Dívida de Defeito	eXcomment	15	19	27	51	112
	DebtHunter	0	0	1	3	4
Dívida de Requisitos	eXcomment	5	2	2	8	17
	DebtHunter	4	0	15	7	26
Dívida de Documentação	eXcomment	0	0	0	0	0
	DebtHunter	0	0	0	0	0
Dívida de Teste	eXcomment	1	0	0	0	1
	DebtHunter	0	0	0	0	0
Total	eXcomment *	38	32	53	110	233
	DebtHunter	9	16	20	42	85

*: A diferença no total de itens de dívida identificados pela eXcomment em cada sistema é dada pelos itens com tipos de dívida não identificados pela DebtHunter.

Analisando estes resultados pondo em perspectiva o tamanho dos sistemas-alvo, observou-se que eXcomment identifica um número superior de itens de dívida técnica por linha de código em relação a DebtHunter. Em média, enquanto a eXcomment identificou 4, 9 itens de dívida a cada 10K linhas de código, a DebtHunter identificou apenas 1, 68 itens. Por outro lado, com a relação ao número de comentários válidos de cada sistema, verificou-se também que eXcomment identificou um número superior de itens de dívida em relação a DebtHunter. Em média, enquanto eXcomment identificou 5, 9 itens de dívida técnica a cada 100 comentários válidos, a DebtHunter identificou apenas 1, 2 itens.

Em resumo, a distribuição das dívidas encontradas pela ferramentas eXcomment e DebtHunter concentram-se em *Dívida de Design, Dívida de Defeito* e *Dívida de Requisitos.*

5 AMEAÇAS À VALIDADE

Em qualquer estudo de caso, especialmente em projetos pequenos, como é o caso deste, ameaças à validade externa são significativas. Em seguida, discute-se algumas ameaças à validade deste estudo, bem como as atitudes tomadas para contornar ou minimizar os efeitos destas.

Validade de conclusão: Infelizmente, a quantidade de sistemas-alvo não é suficiente para uma conclusão generalista. No entanto, buscou-se avaliar os mesmos com isenção, bem como com a revisão da avaliação por um segundo pesquisador.

Validade de construto: a mesma versão de todos os sistemas-alvo foi utilizada pela mesma versão de cada um das ferramentas avaliadas. Além disso, as dívidas apontadas pelas ferramentas não foram validadas à posteriori, pois estava fora do escopo deste trabalho.

Validade de externa: ambas as ferramentas avaliadas são protótipos acadêmicos. Assim, não há garantia de que os resultados apresentados por elas possa ser generalizado. Adicionalmente, não se pode generalizar as observações apresentadas para outros contextos de desenvolvimento, tampouco outros domínios de software.

6 CONCLUSÃO E TRABALHOS FUTUROS

Neste estudo investigou-se o comportamento de duas ferramentas de identificação de dívida técnica auto-admitidas que utilizam a mineração de comentários de código-fonte quanto a intersecção de suas classificações e da distribuição dos itens de DT classificados. Os resultados indicaram que, dos itens de dívida técnica classificados pela eXcomment, a DebtHunter também classifica, em média, apenas 7% deles. Por outro lado, dos itens de dívida classificados pela DebtHunter, a eXcomment também classifica, 19, 9% deles. Dado os resultados discutidos, é possível observar que são necessários mais estudos acerca da identificação de dívida técnica através de comentários de código-fonte, ou mesmo, a validação manual das dívidas técnicas existentes nos sistemas analisados. Percebeu-se ainda que, apesar da discordância entre eXcomment e DebtHunter quanto à quantidade de itens de DT presente em cada um dos sistemas-alvo, as ferramentas parecem convergir quanto aos tipos *Dívida de Design*, *Dívida de Defeito* e *Dívida de Requisitos*. E ainda que, houve pouca ou nenhuma classificação de *Dívida de Documentação* e *Dívida de Teste* pelas ferramentas. Pelo melhor de nosso conhecimento, não há na literatura valores de referência para esta interseção que pudessem nos ajudar a julgar se estes valores encontrados representam algo positivo ou se representam características inerentes das técnicas diferentes utilizadas pelas ferramentas.

O estudo aqui apresentado pode ser estendido para obter-se mais confiança nos resultados aqui discutidos. Como trabalhos futuros, é possível investigar o comportamento das ferramentas em mais sistemas-alvo de outros domínios e tamanhos para a construção de evidências da intersecção dos resultados. A avaliação de outras ferramentas disponíveis na literatura com o mesmo propósito também se faz necessário. Pretende-se ainda realizar a construção de um oráculo para verificar a concordância das ferramentas com um oráculo formado por desenvolvedores, além da utilização de métricas conhecidas para a validação da eficácia das ferramentas.

DADOS DO ESTUDO

Todos os dados coletados durante este estudo podem ser encontrados em: https://doi.org/10.5281/zenodo.6857710

REFERÊNCIAS

[1] Nicolli S. R. Alves, Manoel Gomes de Mendonça Neto, and Rodrigo Oliveira Spínola. 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information Software Technologies* 102 (2018), 117–145. https://doi.org/10.1016/j.infsof.2018.05.010

[2] Paris C. Avgeriou, Davide Taibi, Apostolos Ampatzoglou, Francesca Arcelli Fontana, Terese Besker, Alexander Chatzigeorgiou, Valentina Lenarduzzi, Antonio Martini, Athanasia Moschou, Ilaria Pigazzini, Nyyti Saarimaki, Darius Daniel Sas, Saulo Soares de Toledo, and Angeliki Agathi Tsintzira. 2021. An Overview and Comparison of Technical Debt Measurement Tools. *IEEE Software* 38, 3 (2021), 61–71. https://doi.org/10.1109/MS.2020.3024958

[3] Gabriele Bavota and Barbara Russo. 2016. A Large-Scale Empirical Study on Self-Admitted Technical Debt. In *Proceedings of the 13th International Conference on Mining Software Repositories* (Austin, Texas) *(MSR '16)*. Association for Computing Machinery, New York, NY, USA, 315–326. https://doi.org/10.1145/2901739.2901742

[4] Mário André de Freitas Farias, Manoel Gomes de Mendonça Neto, Marcos Kalinowski, and Rodrigo Oliveira Spínola. 2020. Identifying self-admitted technical debt through code comment analysis with a contextualized vocabulary. *Information Software Technologies* 121 (2020), 106270. https://doi.org/10.1016/j.infsof.2020.106270

[5] Felipe Gustavo de S. Gomes, Thiago Souto Mendes, Rodrigo O. Spínola, Manoel Mendonça, and Mário Farias. 2019. Uma análise da relação entre code smells e dívida técnica auto-admitida. In *Anais do VII Workshop on Software Visualization, Evolution and Maintenance (VEM)* (Salvador). SBC, Porto Alegre, RS, Brasil, 37–44. https://doi.org/10.5753/vem.2019.7582

[6] E. D. S. Maldonado and E. Shihab. 2015. Detecting and quantifying different types of self-admitted technical debt.. In *Proceedings of the 7th Workshop on Managing Technical Debt* (Waikiki, Honolulu, HI, USA) *(MTD '11)*. IEEE, New York, NY, USA, 9–15. https://doi.org/10.1109/MTD.2015.7332619

[7] Isabela Oliveira, Humberto Marques-Neto, and Laerte Xavier. 2020. Analisando Estratégias para Identificação de Dívidas Técnicas. In *Anais do VIII Workshop de Visualização, Evolução e Manutenção de Software* (Evento Online). SBC, Porto Alegre, RS, Brasil, 9–16. https://doi.org/10.5753/vem.2020.14523

[8] Aniket Potdar and Emad Shihab. 2014. An exploratory study on self-admitted technical debt. In *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 91–100.

[9] Irene Sala, Antonela Tommasel, and Francesca Arcelli Fontana. 2021. DebtHunter: A Machine Learning-Based Approach for Detecting Self-Admitted Technical Debt. In *Evaluation and Assessment in Software Engineering* (Trondheim, Norway) *(EASE 2021)*. Association for Computing Machinery, New York, NY, USA, 278–283. https://doi.org/10.1145/3463274.3464455

[10] Nico Zazworka, Rodrigo O. Spínola, Antonio Vetro', Forrest Shull, and Carolyn Seaman. 2013. A Case Study on Effectively Identifying Technical Debt *(EASE '13)*. Association for Computing Machinery, New York, NY, USA, 42–47. https://doi.org/10.1145/2460999.2461005