A Catalog of Prevention Strategies for Test Technical Debt

Levi Almeida da Silva
levi.silva@aluno.uece.br
Universidade Estadual do Ceará
Fortaleza, Ceará, Brasil

Ismayle de Sousa Santos
ismalyle.santos@uece.br
Universidade Estadual do Ceará
Fortaleza, Ceará, Brasil

RESUMO

Background: The demand for software innovation pressures organizations to balance meeting deadlines with maintaining quality. This situation can lead to Technical Debt (TD), which refers to postponed development tasks that offer short-term gains but harm long-term quality. During the development process, the Testing phase also accumulates TD, known as Test Debt, which affects its effectiveness. **Objective**: To propose a strategy that supports preventive activities for Test Debts. **Method**: A literature review was conducted on Test Debts and their prevention strategies, establishing a relationship between these elements. **Results**: A catalog of 47 categories of strategies to prevent Test Debts was developed. Experts assessed the catalog and praised its clarity and usefulness in preventing Test Debts. **Conclusion**: The proposed catalog emerges as a tool that can assist professionals in decision-making to prevent the occurrence of Test Debts.

CCS CONCEPTS

• **Software and its engineering → Software development techniques**; **Software development methods**.

KEYWORDS

Test Debt. Test. Technical Debt. Prevention

ACM Reference Format:
Levi Almeida da Silva and Ismayle de Sousa Santos. 2024. A Catalog of Prevention Strategies for Test Technical Debt . In *XXIII Brazilian Symposium on Software Quality (SBQS 2024), November 5–8, 2024, Salvador, Brazil.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3701625.3701692

1 INTRODUÇÃO

A demanda por inovações em software tem gerado desafios para as organizações de desenvolvimento de software, especialmente em relação ao cumprimento de prazos e padrões de qualidade. Dessa forma, a necessidade de entregar novos produtos ou funcionalidades pode exigir ajustes devido à falta de tempo, ou outros fatores. Esses ajustes podem resultar em "Dívidas Técnicas"(DT).

Conforme conceituado por Cunningham [16], a DT denota a porção do processo de desenvolvimento de software que não foi entregue em conformidade com o planejado, implicando, consequentemente, que essa discrepância deva ser tratada em uma etapa subsequente do ciclo de disponibilização do software.

Dentro desse desenvolvimento, Primão *et al.* [34] reforçam que o processo de Teste representa uma etapa intrínseca e relevante no ciclo de desenvolvimento de software, pois desempenha um papel crucial na promoção da maturidade de um sistema. Segundo Roger e Bruce [39], em contextos de empreendimentos de software, a escassez de recursos temporais ou materiais leva as instituições a comprometer os procedimentos de garantia de qualidade. Esse tipo de comportamento também resulta na "Dívida Técnica", que no campo de Teste de software é chamado de "Débito de Teste". Essa forma de Dívida Técnica ocorre devido a decisões inadequadas nas atividades de teste, como falta de testes adequados e estimativas incorretas, como destaca por Samarthya *et al.* [41].

Diante desse cenário, objetiva-se a mitigação da ocorrência de DT mediante a implementação de práticas de gerenciamento. De acordo com Li *et al.* [29], as práticas de gerenciamento podem abordar oito fundamentos relacionados às Dívidas Técnicas, são elas: (i) Prevenção; (ii) Identificação; (iii) Medição; (iv) Priorização; (v) Monitoramento; (vi) Reembolso/Pagamento; (vii) Documentação e (viii) Comunicação.

Conscientes dessas bases, Pérez *et al.*[32] sustentam que técnicas preventivas desempenham um papel crucial na mitigação das Dívidas Técnicas, o que é corroborado por Rios *et al.* [36] ao observar que a prevenção de Dívidas pode ser mais eficaz e econômica para a equipe de desenvolvimento do que a posterior liquidação.

Assim, o problema investigado nesse trabalho é: ***Como as equipes de teste podem proativamente mitigar o Débito de Teste sem depender apenas de ações corretivas?***

Conforme Aragão *et al.* [7], a maioria dos estudos se concentra apenas em algumas causas específicas de Débito de Teste (por exemplo, falta de testes automatizados). Além disso, os autores afirmam que não foram identificadas ferramentas ou metodologias que abordassem de maneira abrangente a prevenção da DT para esse tipo de dívida.

Sendo assim, esta pesquisa tem como objetivo principal o desenvolvimento de um catálogo de estratégias de prevenção de Dívidas Técnicas no contexto de teste de software, elaborando um relacionamento entre ações de prevenção e Dívidas de Teste. Para isso, têm-se como objetivos secundários: identificar Dívidas de Testes e identificar estratégias de Prevenção de Dívidas Técnicas para o estabelecimento dessas relações.

Ao abordar esta temática, a presente pesquisa contribui para o levantamento do estado da arte de prevenção de DT, além de conferir diretrizes para a mitigação de Dívidas de Teste. Nesse sentido, através do levantamento das situações que podem gerá-las e como preveni-las, pode-se potencialmente garantir a entrega do sistema com qualidade e a realização eficaz dos testes.

SBQS 2024, November 5–8, 2024, Salvador, Brazil

ACM ISBN 979-8-4007-1777-2/24/11
https://doi.org/10.1145/3701625.3701692

2 DÍVIDA TÉCNICA

Por seu trabalho, Cunningham [16] é amplamente reconhecido como o pioneiro no campo da Dívida Técnica. Ele é creditado por introduzir esse conceito na Engenharia de Software. É importante notar que, em sua visão, a DT estava inicialmente associada apenas ao código, enfatizando que quanto mais tempo a DT permanecer sem pagamento, mais dispendiosa será sua resolução.

Também é destacado pelo o autor a perspectiva de que, a curto prazo, adquirir DT pode ser vista como uma medida positiva, desde que se tenha a visão de que, no futuro, será necessário quitar essa dívida para evitar seu acúmulo. Essa situação pode ocorrer, por exemplo, quando é necessário recorrer a DT devido a código mal escrito para atender a prazos rigorosos de entrega.

Conforme a abordagem de Brown *et al.* [13], é possível conceber a DT como uma ferramenta para caracterizar a discrepância entre o estado presente de um sistema de software e um estado hipotético considerado "ideal". Nesse contexto, à medida que se delineia a visão do destino almejado e se avalia o ponto de partida, decisões são tomadas para direcionar os esforços rumo ao estado ideal.

Contrapondo a perspectiva centrada no código quando se trata de (DT), Klinger *et al.* [27] destaca que essa não se restringe exclusivamente aos aspectos técnicos ou à programação. Ela transcende esses limites e é influenciada por uma variedade de fatores, sejam eles de natureza técnica ou não. Assim, os tipos de DT variam de acordo com a atividade de desenvolvimento de software relacionado. Segundo da Fonseca Lage [17], por exemplo, o Débito de Teste pode ser caracterizado como problemas identificados durante as atividades de Teste, demonstrando potencial impacto na qualidade dessas atividades.

Cabe ressaltar que o teste de software, conforme Myers *et al.* [31], se configura como um processo planejado, com o propósito de certificar a conformidade do código de computador com sua finalidade projetada, assegurando simultaneamente que não introduza comportamentos não intencionais. A exemplificação de Débitos de Teste inclui então situações como testes planejados que não foram executados e deficiências conhecidas no conjunto de testes, como a baixa cobertura de código.

3 TRABALHOS RELACIONADOS

O trabalho de Aragão *et al.* [9] tem o propósito de apresentar um catálogo de subtipos de Débitos de Teste e atividades de gerenciamento passíveis de aplicação em projetos de desenvolvimento de software. O resultado tangível é um catálogo que se destina a auxiliar profissionais de Teste de software na identificação e gestão das Dívidas de Teste. Entretanto, o trabalho apresentado não aborda especificamente técnicas de prevenção de Débito de Teste. Embora o catálogo apresente atividades de gerenciamento para cada subtipo de Débito de Teste identificada, ele não fornece informações detalhadas sobre como prevenir a criação desse tipo de dívida.

No trabalho de Accioly *et al.*[1], o propósito consistiu em realizar uma pesquisa exploratória de modo que entendesse, por meio de entrevistas, como o gerenciamento de DT está sendo realizado na prática pelos profissionais no desenvolvimento ágil. O artigo ressalta que entender as causas da DT pode ajudar a prevenir tal situação. Dessa forma, o trabalho destaca alguns insights para gerência de DT, como a criação de uma task específica para a DT, adicionando-a ao Backlog para ser solucionada quando possível e uso de plataformas de gerenciamento. Porém, por mais que o artigo mencione que a gestão de DT pode mitigar Dívidas Técnicas não relacionadas ao código, não é comentado diretamente sobre como prevenir Dívidas de Teste.

O propósito do estudo de Pérez [32] consiste em investigar as práticas adotadas pelos arquitetos de software no que tange ao gerenciamento e à prevenção de Dívida Técnica em projetos de desenvolvimento de software. Os resultados indicam que a Dívida Técnica ligada ao Teste é uma preocupação em projetos de desenvolvimento de software. Porém, o estudo destaca que as práticas preventivas mais adotadas pelos arquitetos de software para mitigar a ocorrência de DT estão relacionadas à revisão de código, definição de escopo/requisitos e arquitetura/design. Sendo assim, o trabalho não oferece detalhes específicos sobre estratégias de gerenciamento de Dívidas de Teste, concentrando-se primariamente nas fases de desenvolvimento do software.

Além disso, embora existam trabalhos que abordem prevenção de dívidas técnicas de maneira geral, por não trazerem uma visão atrelada as características da área de testes, não permitem aos engenheiros uma seleção das estratégias mais relevantes para seu contexto de desenvolvimento de software.

4 METODOLOGIA

A abordagem metodológica adotada neste trabalho, apresentada na Figura 1, foi dividida em duas fases, Concepção e Avaliação, assim como o de Dias-Neto *et al.*[20]. Na fase de Concepção, foram realizadas as seguintes etapas: (i) Revisão de literatura, (ii) Identificação de Dívidas de Teste a partir dos trabalhos identificados na literatura, (iii) Identificação de estratégias de prevenção de Dívidas Técnicas a partir das pesquisas encontradas, (iv) Criação dos relacionamentos entre Dívidas de Teste e suas respectivas prevenções. Já na fase de Avaliação, foram conduzidas as etapas: (v) Desenvolvimento do questionário para avaliadores e (vi) Avaliação das respostas.

4.1 Concepção

Na fase inicial da pesquisa, foi realizada uma revisão extensa da literatura, focando na análise de estudos, trabalhos acadêmicos e publicações relevantes sobre dívidas técnicas em testes de software. Nesse contexto, conduziu-se uma revisão de literatura nas plataformas ACM[1], IEEE[2] e Google Scholar[3] (no Google Scholar, analisaram-se os resultados até a quinta página devido ao tempo disponível. Foram selecionados artigos a partir de 2019 para garantir uma visão atualizada, utilizando strings de busca específicas para identificar estudos sobre Débito de Teste e a prevenção de Dívida Técnica. A escolha desse recorte temporal se baseia no trabalho de Aragão *et al.* [9], que apontou a inexistência, até aquele momento, de ferramentas ou processos que oferecessem suporte à prevenção de dívidas relacionadas a testes. Além disso, utilizou-se a ferramenta Parsifal[4] para o gerenciamento dos artigos.

Ademais, devido à quantidade substancial de artigos retornados, foram conduzidas investigações utilizando palavras-chave no

[1]https://dl.acm.org/
[2]https://ieeexplore.ieee.org/Xplore/home.jsp/
[3]https://scholar.google.com/
[4]https://parsif.al/

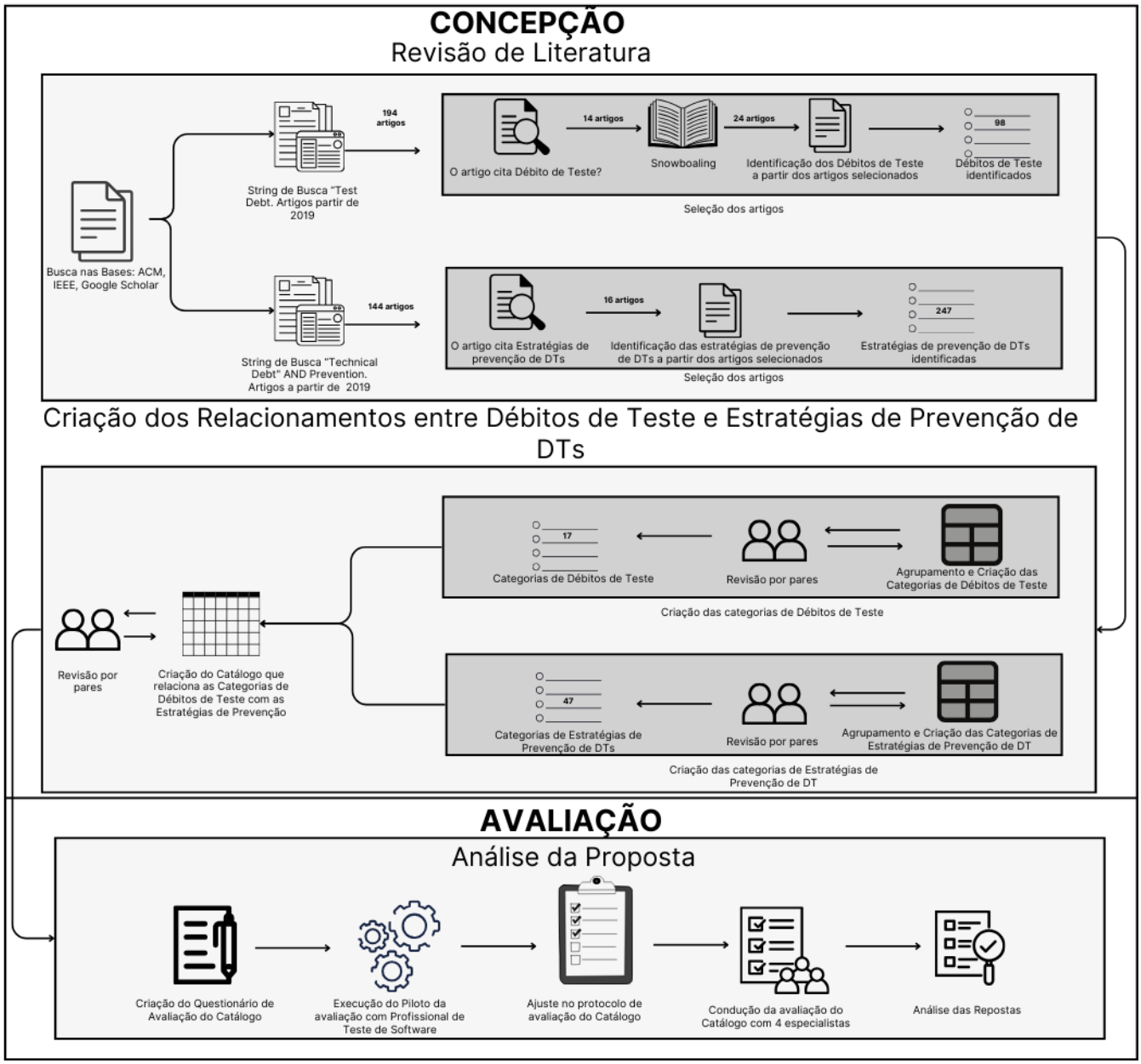

Figura 1: Representação da abordagem metodológica aplicada

documento, buscando identificar conteúdos que atendessem aos critérios de aceitação discorridos anteriormente.

Assim, foi utilizada a string de busca "*Test Debt*"nas plataformas previamente mencionadas, seguindo o critério de data mencionado anteriormente. Após a seleção dos artigos das bases, adotou-se a técnica de *snowballing forward* a fim de encontrar mais débitos de testes. Essa técnica foi aplicada a partir dos artigos aceitos. O critério para considerar um artigo como aceito foi uma resposta afirmativa à seguinte pergunta: ***"O artigo menciona e fornece exemplos de Débitos de Teste?"***.

Durante a análise dos artigos retornados, identificou-se a presença de duplicatas. Entretanto, somente após a aplicação da técnica de *snowballing forward*, foi realizada a verificação de duplicidade por meio da funcionalidade "*Find duplicates*" da ferramenta Parsifal. Como resultado, 24 artigos foram selecionados.

Além disso, os critérios para rejeição de um artigo incluíram: ausência de menção direta aos Débitos de Teste com exemplos, artigos duplicados, ou indisponibilidade para leitura devido a custos para comprar o artigo.

Além disso, empregou-se a string de busca "*"Technical Debt"AND Prevention*" nas plataformas previamente mencionadas, seguindo o critério de data mencionado anteriormente. Porém, durante a escolha desses artigos, não aplicou-se a técnica de *snowballing forward*, pois o número de estratégias de prevenção foi elevado e abrangente. O critério para considerar um artigo como aceito foi a resposta afirmativa à seguinte pergunta: ***"O artigo menciona e fornece exemplos de estratégias de prevenção de dívidas técnicas?"***.

Durante a análise dos artigos retornados, identificou-se a presença de duplicatas, não só no próprio resultado, mas também em relação aos artigos que apareceram na primeira busca.

Além disso, os critérios para rejeição de um artigo incluíram: ausência de menção direta à estratégias de prevenção de dívidas técnicas, artigos duplicados, ou indisponibilidade para leitura devido a custos para comprar o artigo. Como resultado, foram selecionados 16 artigos.

4.1.1 Criação dos Relacionamentos. Ademais, na etapa de Criação dos relacionamentos entre Dívidas de Teste e suas respectivas prevenções, percebeu-se que pesquisa relacionada à identificação de Débitos de Teste, resultou na identificação de um total de **98 Dívidas de Testes**. Vale ressaltar que algumas dessas apresentaram duplicidade, sendo, portanto, consolidadas, traduzidas e agrupadas em 17 categorias de acordo com suas semelhanças, conforme a Tabela 1.

A título de ilustração, o estudo conduzido por Li *et al.* [28] menciona a existência de um Débito de Teste denominado "Falta de testes", ao passo que a pesquisa conduzida por Das *et al.* [18] aborda um Débito de Teste designado como "Falta de testes estruturados". Nesse contexto, ambos os exemplos foram agrupados na categoria "Falta de testes"(ID Categoria Débito de Teste 12) devido à sua semelhança. As demais categorias seguiram o mesmo princípio.

Adicionalmente, na Tabela 1, encontra-se uma Visão Geral que descreve cada categoria de Débito de Teste, elaborada com base nos artigos analisados. É importante observar que nem todos os artigos selecionados forneceram uma definição explícita para cada Débito de Teste. Por exemplo, no artigo de Aragão *et al.*[7], é mencionado que "Adiamento de Teste"ocorre quando um teste planejado para determinado release é postergado para outro momento.

Também foi identificado um conjunto total de **247 estratégias destinadas à prevenção de Dívidas Técnicas**. É importante destacar que algumas destas manifestaram-se de maneira duplicada ou com significados semelhantes, o que motivou uma consolidação, tradução e agrupamento em 47 categorias, conforme apresentado na Tabela 2.

A título de ilustração, o estudo conduzido por Berenguer *et al.* [10] menciona a existência de uma estratégia de prevenção denominada "Fornecer Treinamento", ao passo que a pesquisa conduzida por Perez *et al.*[32] aborda uma estratégia de prevenção designada como "Treinamento". Nesse contexto, ambos os exemplos foram agrupados na categoria "Treinar adequadamente os membros da equipe"(ID Categoria de prevenção 44) devido à sua semelhança. As demais categorias seguiram o mesmo princípio.

No software *Google Sheets*, o relacionamento entre os Débitos de Teste e as estratégias de prevenção foi representado por meio de uma matriz, na qual os Débitos de Teste foram listados nas colunas e as estratégias de prevenção nas linhas, conforme visualizado na Tabela 4. Essa matriz possui um total de 799 células, preenchidas ou não. Os relacionamentos, quando presentes, foram inicialmente obtidos através de citações explícitas dos artigos lidos.

Adicionalmente, os outros relacionamentos foram propostos com base nas definições de Débitos de Teste e Estratégias de prevenção e experiência dos pesquisadores envolvidos, por exemplo, a indicação de que a Estratégia de prevenção "Alocar Profissionais Qualificados na equipe", pode prevenir o Débito de Teste "Adiamento de Teste. Esse processo se deu através de uma revisão por pares para que não prevalesse o viés de um dos autores. Assim, o resultado desses relacionamentos deu origem à Tabela 4.

A Tabela 3 apresenta os relacionamentos explícitos oriundos de citação e os relacionamentos propostos. Nessa tabela, a abreviação "CIT"indica que o relacionamento foi estabelecido a partir de menções explícitas nos artigos selecionados na revisão, enquanto "PROP"implica que a relação foi proposta com base nas definições apresentadas de dívidas técnicas e das estratégias de prevenção. É importante ressaltar que a tabela foi avaliada em pares para verificar o nível de concordância dos avaliadores em relação ao Catálogo.

Já a ausência de conteúdo na célula, implica que aquela estratégia de prevenção não evita tal Débito de Teste. Na Figura 2 por exemplo, estratégia de prevenção "Alocar adequadamente as tarefas entre a equipe"não previne o Débito de Teste "Baixa cobertura de Código".

4.2 Avaliação

A fim de coletar informações sobre os respondentes, elaborou-se um formulário, utilizando *Google Forms* destinado a coletar as percepções dos avaliadores quanto à utilidade e à coerência dos relacionamentos estabelecidos. O questionário compreendia as seguintes seções: (i) Dados pessoais e experiência em testes de software, (ii) Experiência com Débitos de Teste e suas prevenções, (iii) Avaliação Quantitativa da Matriz e (iv) Avaliação Qualitativa da Matriz.

Para a condução da avaliação, o formulário e a matriz de avaliação, no formato de planilha, foram enviados por e-mail a quatro avaliadores. Os avaliadores incluíam um graduando, um mestrando, um mestre e um doutorando, todos com experiência na área de qualidade de software. Vale destacar que 50% dos participantes, em uma escala Likert de 1 a 5 - onde 1 indicava nenhuma familiaridade e 5 alta familiaridade com o conceito de Dívida Técnica (DT) - atribuíram nota 5, demonstrando profundo conhecimento sobre o tema. Além disso, todos os avaliadores já tiveram experiência direta com DT e precisaram tomar medidas preventivas em relação a ela. Outro ponto relevante é que 75% dos participantes possuem entre 2 e 5 anos de experiência com Teste de Software.

A planilha enviada consistiu em três abas: a primeira dedicada às instruções para o preenchimento da matriz, a segunda contendo a própria matriz (Tabela 4) e a terceira apresentando uma visão geral dos débitos de teste. A indicação de um relacionamento entre a estratégia de prevenção e o débito de teste era feita apenas pela marca "X".

O preenchimento tanto do questionário quanto da simulação de preenchimento da matriz foi submetido a uma avaliação piloto conduzida por um graduando em Ciência da Computação com 2 anos de experiência em teste de software. A finalização do questionário demandou, em média, 4 minutos, variando de acordo com

Tabela 1: Categorias de Débitos de Teste

ID	Categoria de Débito de Teste	Descrição	Artigos
1	Adiamento de testes	Ocorre quando, por alguma questão (e.g., diminuição do escopo de teste e pressão para entrega antecipada), um teste planejado para aquele release deve ser realizado em outro momento	[8], [7], [9]
2	Alocação inadequada da equipe	Ocorre quando, por alguma questão (e.g., demora para entregar a funcionalidade para teste, acúmulo de outras responsabilidade), o testes demoram a serem executados ainda dentro do mesmo release	[9],[8],
3	Atraso na realização de testes	Ocorre quando, por alguma questão (e.g., demora para entregar a funcionalidade para teste, acúmulo de outras responsabilidade), o testes demoram a serem executados ainda dentro do mesmo release	[25]
4	Baixa cobertura de código	Ocorre quando os testes unitários previstos não abrangem uma boa porção do código-fonte	[12], [2],[28], [19], [26], [14], [42]
5	Baixa cobertura de testes	Ocorre quando os testes previstos não abrangem toda aquela funcionalidade e seus impactos	[44]
6	Defeitos não encontrados em testes	Ocorre quando os Defeitos existentes não foram encontrados nos testes realizados, assim, fazendo que esses defeitos tenham que ser tratados em outros momentos	[3], [9], [8], [7]
7	Equipamento inadequado para realização de testes	Ocorre quando o equipamento utilizado para realizar o teste não é adequado para aquele tipo de teste ou não suporta as ferramentas para os testes ou gera lentidão nos testes	[7]
8	Erro de estimativa de teste	Ocorre quando o testador estima determinado tempo para realizar um teste, porém leva mais tempo que o necessário, a ponto de comprometer o planejamento inicial	[9], [8], [7]
9	Erros de execução de teste	Ocorre quando o teste planejado é executado erroneamente, assim, sendo necessário, posteriormente, executar novamente o teste da maneira correta para validar a funcionalidade	[9]
10	Falta de automação de testes	Ocorre quando não há presença de testes automatizados e o testador ter que fazê-los manualmente, levando mais tempo para validar a funcionalidade	[2], [3], [45], [18],[9], [8]
11	Falta de interpretação de teste	Ocorre quando um teste é mal escrito ou não é entendido pela pessoa que irá testar aquela funcionalidade, o que pode implicar em tempo para compreender como executar o teste ou uma má realização	[12]
12	Falta de testes	Ocorre quando não é previsto realização de testes para determinado contexto	[40], [12],[2],[38]
13	Falta de testes unitários	Ocorre quando há carência de testes unitários	[45], [14], [15]
14	Testes incompletos	Ocorre quando o teste iniciou, mas não foi finalizado a tempo por qualquer fator	[25]
15	Testes caros	Ocorre quando os testes são dispendioso resultando em desaceleração das atividades de teste	[28], [19], [26], [43], [44],[9], [8]
16	Teste inadequado	Ocorre quando o teste planejado não é adequado para aquele tipo de situação, assim tendo reavaliar o contexto para planejar um teste coerente	[12], [35]
17	Testes interrompidos	Ocorre quando os testes unitários não são mais utilizados de forma adequada no código por decisão humana	[15], [25]

as respostas abertas. Já o preenchimento da matriz, por outro lado, demandou, em média, 40 minutos.

Os participantes foram solicitados a expressar sua concordância, discordância ou fornecer sugestões de relacionamento (para as células que não indicavam relação) por meio de cores. Nesse contexto, a cor vermelha indicava discordância para àquela relação, e a amarela sinalizava a percepção de que aquela estratégia de prevenção evitava também o respectivo débito de teste não indicado pelo autor. Assim, caso o avaliador concordasse com o relacionamento,

Tabela 2: Categorias de Estratégias de Prevenção de Dívidas Técnicas

ID	Estratégias de Prevenção de Dívidas técnicas	Artigos
1	Adotar boas práticas de testes	[22], [10], [33], [32]
2	Alocar adequadamente as tarefas entre a equipe	[22], [10], [33]
3	Alocar Profissionais Qualificados na equipe	[10]
4	Alocar recursos na equipe com um todo	[22], [11]
5	Analisar o custo-beneficio do teste	[11]
6	Analisar profundamente as funcionalidades envolvidas	[11]
7	Analisar risco e impactos dos testes (não necessariamente envolvendo custos)	[22], [24], [11]
8	Automatizar tarefas (não necessariamente testes)	[22], [21]
9	Considerar restrições técnicas	[10], [11]
10	Criar testes automatizados	[22], [21]
11	Criar testes para as novas funcionalidades	[10], [22]
12	Entender o processo de desenvolvimento da empresa	[11]
13	Escrever bons casos de testes de integração	[46]
14	Escrever bons casos de testes de sistema	[46]
15	Escrever bons casos de testes unitários	[46]
16	Fornecer os requisitos mais cedo	[11]
17	Implementar ações preventivas de dívidas técnicas	[22]
18	Localizar e corrigir bugs com rapidez e precisão	[46], [47]
19	Manter as documentações pertinentes atualizadas e bem definidas	[11]
20	Melhorar estimativa de esforço de teste	[11],
21	Melhorar planejamento de projeto	[11]
22	Monitorar e controlar atividades do projeto	[22], [23]
23	Organizar repositório do projeto	[22], [23]
24	Pagar dívidas técnicas	[11]
25	Remover tarefas de baixa prioridade	[11]
26	Ter ambientes para testes	[11]
27	Ter atividades padronizadas	[11]
28	Ter claro a Definição de pronto	[6], [4], [5]
29	Ter comprometimento com sua responsabilidade	[11], [30]
30	Ter conhecimento técnico	[11]
31	Ter conscientização e gerenciamento de Dívidas Técnicas	[11]
32	Ter integração contínua (CI)	[22], [11]
33	Ter métricas de testes bem definidas	[11]
34	Ter padrões de código	[6], [22], [2], [32], [11]
35	Ter prazos bem definidos	[11], [22], [10]
36	Ter reuniões de retrospectiva ou Ter reuniões para analisar causa e consequência de dívidas técnicas	[6], [2], [5], [37]
37	Ter um código bem escrito	[11]
38	Ter uma arquitetura do sistema bem definida	[32], [22] [11]
39	Ter uma boa comunicação com a equipe	[32], [11], [24]
40	Ter uma cobertura de testes adequada	[32], [11]
41	Ter uma definição clara do escopo de trabalho	[30], [22]
42	Ter uma equipe organizada	[11]
43	Ter uma governança em TI	[11]
44	Treinar adequadamente os membros da equipe	[32], [11], [24], [10], [10], [22]
45	Usar diversas estratégias de testes	[11]
46	Usar interações curtas de feedback	[23]
47	Utilizar as lições aprendidas para uma próxima etapa	[11]

apenas deixava o X na célula. Caso concordasse com a ausência do relacionamento, deixaria a célula vazia.

Por exemplo, na Figura 2, é apresentada a representação gráfica do preenchimento da matriz pelos avaliadores. Nessa figura, um dos respondentes expressou concordância ao afirmar que a estratégia "Adotar boas práticas de testes"está relacionada ao débito de teste "Adiamento de testes". Contudo, discordou que a estratégia "Alocar adequadamente tarefas entre a equipe"se relaciona com o débito "Adiamento de testes". Além disso, indicou que a estratégia "Adotar boas práticas de testes"tem relação com o débito "Alocação inadequada da equipe". Percebe-se que quando é adicionada cor em uma célula, no caso a vermelha, o "X"não permanece.

Tabela 3: Catálogo com seus tipos de relacionamentos

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	PROP		PROP	CIT		PROP			CIT		PROP	PROP	CIT	PROP	PROP	PROP	CIT
2	CIT	PROP	CIT					CIT						CIT			
3	PROP	CIT	PROP	PROP	PRPO	PROP		PROP	PROP	PROP	PROP	PROP	PROP	PROP		PROP	PROP
4	PROP	CIT	PROP				CIT							PROP			
5					PROP		PROP							PROP	CIT		
6	PROP		PROP		PROP	PROP		PROP	PROP		PROP	PROP				PROP	
7	CIT		CIT		CIT	CIT						PROP			PROP	PROP	
8	PROP		PROP		PROP			PROP		CIT							
9								CIT	PROP		PROP					PROP	
10	PROP		PROP		PROP	PROP		PROP	PROP	CIT	PROP	PROP				PROP	
11				PROP	PROP	PROP						CIT	PROP				
12	PROP		PROP		PROP	PROP			PROP			PROP					PROP
13	PROP		PROP		PROP	PROP		PROP	PROP		PROP	PROP				PROP	
14	PROP		PROP		PROP	PROP		PROP	PROP		PROP	PROP				PROP	
15				CIT									CIT				CIT
16	PROP		PROP		PROP			PROP	PROP		PROP	PROP				PROP	
17	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP
18									PROP			PROP					
19	PROP		PROP		PROP	PROP		PROP	PROP		PROP	PROP		PROP			
20	PROP		PROP					CIT									
21	PROP		PROP					PROP			PROP	CIT	PROP		PROP		
22	PROP		PROP			PROP		PROP				PROP	PROP		PROP		PROP
23				PROP									PROP				CIT
24	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP
25	PROP		PROP											PROP			
26	PROP		PROP			PROP						PROP		PROP	PROP	PROP	
27	PROP		PROP			PROP		PROP			PROP	PROP					
28	PROP		PROP	PROP	PROP	PROP		PROP	PROP		PROP			PROP		PROP	
29	PROP		PROP	PROP	PROP	PROP		PROP	PROP		PROP	PROP	PROP	PROP		PROP	
30	PROP		PROP	PROP	PROP	PROP		PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP
31	PROP	PROP	CIT	PROP	PROP	PROP	PROP	PROP	PROP			CIT	PROP	PROP	PROP	PROP	
32				CIT									CIT				CIT
33	PROP		PROP	PROP	PROP	PROP		CIT					PROP		PROP		
34				CIT						PROP			CIT				CIT
35	PROP		PROP									PROP	PROP	PROP			
36	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP
37				CIT						PROP			CIT				CIT
38				CIT						PROP			CIT				CIT
39	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP		PROP	PROP	PROP	PROP	PROP		PROP	
40	CIT		CIT		CIT	CIT				PROP		CIT	CIT	PROP	PROP	CIT	
41	PROP		PROP		CIT	PROP		PROP	PROP	PROP	PROP	PROP		PROP	PROP	PROP	
42	PROP	PROP	PROP		PROP						PROP	PROP	PROP				
43	PROP		PROP										PROP				PROP
44	PROP	PROP	PROP	PROP	PROP	PROP		PROP	PROP	PROP	PROP	CIT	CIT	PROP		PROP	PROP
45	CIT		CIT	PROP	CIT	PROP				PROP		CIT	PROP	PROP	PROP	PROP	
46	PROP		PROP	PROP	PROP	PROP		PROP	PROP	PROP	PROP	PROP	PROP			PROP	PROP
47	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP	PROP

Para a análise das respostas, foram examinadas as percepções dos avaliadores com base nos dados fornecidos no formulário. Adicionalmente, procedeu-se a uma comparação entre as proposições do autor sobre os relacionamentos entre estratégias de prevenção de dívidas técnicas e débitos de testes e as percepções dos avaliadores.

Para a avaliação, as planilhas foram examinadas no software *Excel*, utilizando a função de localização de informações formatadas, considerando a cor específica como critério de busca. A análise objetivou determinar o número exato de relacionamentos com os quais o avaliador concordou, contando o número de células não coloridas da planilha. Além disso, para identificar a quantidade de células em que o avaliador discordou da relação, seja por não concordar com a proposta de relação (indicação pela cor vermelha) ou por sugerir uma relação entre os elementos (indicação pela cor

Figura 2: Ilustração do preenchimento da Matriz

	Cores			
DÍVIDAS TÉCNICAS DE TESTE / ESTRATÉGIAS DE PREVENÇÃO	Adiamento de testes	Alocação inadequada da equipe	Atraso na realização de testes	Baixa cobertura de código
Adotar boas práticas de testes	X		X	X
Alocar adequadamente as tarefas entre a equipe		X	X	
Alocar Profissionais Qualificados na equipe	X	X	X	X

amarela) que não estavam relacionados, contou-se o número de células com essas cores e subtraiu-se do valor total de células da Matriz.

5 CATÁLOGO DE ESTRATÉGIAS DE PREVENÇÃO E SUAS RELAÇÕES COM AS DÍVIDAS DE TESTE

Na Tabela 4 encontra-se uma representação da **Matriz de Relacionamentos entre estratégias de Prevenção de Dívidas Técnicas e Dívidas de Teste**. Vale ressaltar que as linhas numeradas correspondem às categorias de estratégias de prevenção de Dívidas Técnicas, conforme detalhadas na Tabela 2, enquanto a numeração das colunas refere-se as categorias de Dívidas de Teste apresentados na Tabela 1. Percebe-se que algumas estratégias de prevenção se destacam, por exemplo, a categoria de estratégia de prevenção ID 47 - "Utilizar as lições aprendidas para uma próxima etapa", pois pode prevenir todas as categorias de Débitos de Teste catalogados. Além disso, o Débito de Teste ID 1 - "Adiamento de testes", é um dos Débitos de Teste com mais variedade de Estratégias de Prevenção.

Conforme evidenciado, a aplicação do Catálogo pode ser delineado da seguinte maneira: Se o usuário decidir empregá-lo com o intuito de determinar quais estratégias de prevenção poderiam potencialmente mitigar um Débito de Teste específico, ele pode identificar, nas colunas, o Débito de Teste de interesse. Posteriormente, procederá à verificação das estratégias de prevenção correspondentes, as quais estarão marcadas com um "X"nas células das colunas relativas ao respectivo Débito de Teste.

6 AVALIAÇÃO DO CATÁLOGO

Os resultados presentes na Tabela 5 e 6 dizem respeito às respostas dos 4 avaliadores, entre pessoas da indústria e pessoas do meio acadêmico. Vale ressaltar que o grupo de avaliadores foi composto por um graduando, um mestrando, um mestre e um doutorando. Além de que todos já lidaram com Dívidas de Teste e com técnicas de prevenção de Dívidas de Teste em seus contextos profissionais.

Para a coleta dos dados, as respostas dos avaliadores foram comparadas com a Matriz contendo os relacionamentos estabelecidos pelo autor. Dessa maneira, o catálogo contou com 799 células disponíveis para representar relacionamentos. Das 799 células, 398 foram preenchidas pelo autor, indicando o relacionamento entre a estratégia de prevenção de Dívida Técnica e o Débito de Teste. Consequentemente, 401 células não conteve nenhum indicativo de relacionamento.

A partir disso, foi analisado o quanto cada avaliador concordou com o relacionamento, existente ou não, entre os elementos, assim como mostra Tabela 5, tendo seus valores em porcentagens aproximados.

Considerando exclusivamente o índice de concordância, esse índice se aproxima de 90%, o que representa um valor relativamente elevado. Tal resultado sugere que os relacionamentos propostos apresentam coerência segundo as experiências dos avaliadores.

Na avaliação qualitativa do catálogo e suas relações, foi empregada uma escala ordinal de cinco pontos para avaliar a utilidade da matriz como ferramenta para orientar a implementação de medidas preventivas contra Dívidas de Teste, a sua intuitividade e a abrangência tanto das estratégias de prevenção quanto das Dívidas de Teste identificadas. Os resultados atribuídos estão presentes na Tabela 6.

A avaliação da Matriz de Relacionamentos revelou uma forte percepção de utilidade, com 75% dos avaliadores concedendo a nota máxima. Quanto à intuição, houve uma divisão equitativa entre notas 4 e 3, indicando uma avaliação intermediária. Em relação à abrangência das estratégias de prevenção e Dívidas de Teste, 50% dos avaliadores atribuíram nota máxima, sugerindo uma cobertura adequada nessas áreas de avaliação. Em resumo, a Matriz é considerada útil na prevenção de Dívidas de Teste, com uma percepção moderada de intuição e uma abrangência equilibrada nas estratégias de prevenção.

Na avaliação qualitativa realizada, os avaliadores observaram, em geral, que a Matriz oferece suporte para a tomada de decisões relacionadas à prevenção de Dívidas de Teste. No entanto, destacaram a necessidade de um conteúdo mais direcionado à explicação ou exemplificação das estratégias de prevenção, devido à percepção de que algumas estratégias podem ser ambíguas.

Por mais que este seja um resultado significativo, a pesquisa pode apresentar ameaças à sua validade devido ao número limitado de avaliadores e à falta de aplicação prática em um contexto de desenvolvimento de software.

6.1 Ameaças à Validade

Devido a natureza empírica do trabalho, cabem ser destacadas algumas ameaças à validade dos resultados obtidos.

Tabela 4: Catálogo

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	x		x	x		x			x		x	x	x	x	x	x	x
2	x	x	x					x						x			
3	x	x	x	x	x	x		x	x	x	x	x	x	x		x	x
4	x	x	x				x							x			
5					x		x							x	x		
6	x		x		x	x		x	x		x	x				x	
7	x		x		x	x						x			x	x	
8	x		x		x			x		x							
9								x	x		x					x	
10	x		x		x	x		x	x	x	x	x				x	
11				x	x	x						x	x				
12	x		x		x	x			x			x					x
13	x		x		x	x		x	x		x	x				x	
14	x		x		x	x		x	x		x	x				x	
15				x									x				x
16	x		x		x			x	x		x	x				x	
17	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
18									x			x					
19	x		x		x	x		x	x		x	x		x			
20	x		x					x									
21	x		x					x			x	x	x		x		
22	x		x			x		x				x	x		x		x
23				x									x				x
24	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
25	x		x											x			
26	x		x			x						x		x	x	x	
27	x		x			x		x			x	x					
28	x		x	x	x	x		x	x		x			x		x	
29	x		x	x	x	x		x	x		x	x	x	x		x	
30	x		x	x	x	x		x	x	x	x	x	x	x	x	x	x
31	x	x	x	x	x	x	x	x	x			x	x	x	x	x	
32				x									x				x
33	x		x	x	x	x		x					x		x		
34				x						x			x				x
35	x		x									x	x	x			
36	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
37				x						x			x				x
38				x						x			x				x
39	x	x	x	x	x	x	x	x		x	x	x	x	x		x	
40	x		x		x	x				x		x	x	x	x	x	
41	x		x		x	x		x	x	x	x	x		x	x	x	
42	x	x	x		x						x	x	x				
43	x		x										x				x
44	x	x	x	x	x	x		x	x	x	x	x	x	x		x	x
45	x		x	x	x	x				x		x	x	x	x	x	
46	x		x	x	x	x		x	x	x	x	x	x			x	x
47	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Quanto as ameaças à **Validade de Construção**, tem-se fatores relacionados as bases utilizadas e as categorias definidas. A utilização de apenas três bases de dados pode não ser suficiente para capturar um espectro amplo de Dívidas de Teste e estratégias de prevenção de DT. No entanto, ela foi realizada em bases amplamente utilizadas na literatura de Computação. Já com relação as categorias definidas para dívidas e estratégias de prevenção, elas foram criadas após a extração das dívidas e estratégias de prevenção. Esta categorização pode ter sofrido um enviesamento no processo.

Tabela 5: Concordância dos Relacionamentos de maneira geral

Avaliador	Grau de Formação	Concordância Geral com os Relacionamentos da Catálogo	
		Itens Concordados	Itens Discordados
A1	Graduando	687 (85,98%)	112 (14,02%)
A2	Mestrando	729 (91,24%)	70 (8,76%)
A3	Mestre	688 (86,11%)	111 (13,89%)
A4	Doutorando	745 (93,24%)	54 (6,76%)
	Total	**89,14%**	**10,86%**

Tabela 6: Valores atribuídos aos critérios de avaliação por cada Avaliador

Avaliador	Grau de Formação	Valor atribuído na escala		
		Utilidade da Matriz	**Intuitividade da Matriz**	**Abrangência da Matriz**
A1	Graduando	5	4	4
A2	Mestrando	4	3	4
A3	Mestre	5	4	5
A4	Doutorando	5	3	5

Cabe ressaltar, no entanto, que elas foram definidas por meio de um processo iterativo de análise das definições extraídas dos artigos selecionados. Além disso, todo o processo passou por revisões por pares.

Quanto à **Validade Interna**, tem-se como ameaça principal o processo de construção da matriz de relacionamentos. Ela foi desenvolvida pelos pesquisadores envolvidos com base nos relacionamentos explícitos nos artigos, nas definições das dívidas e estratégias de prevenção e na experiência dos pesquisadores envolvidos. Para mitigar possíveis vieses no mapeamento, seções de revisões em pares foram realizadas. Além disso, os resultados da avaliação evidenciaram indícios de que o catálogo está consistente e que de fato pode contribuir na mitigação de prevenção de testes. Avaliações futuras com um maior número de especialistas devem fornecer evidências adicionais nesse aspecto. Além disso, planeja-se implementar a codificação nas categorias para refiná-las. Pretende-se também aplicar o catálogo em projetos industriais, por meio de um estudo de caso, com o objetivo de analisar seu uso prático.

Com relação as ameaças à **Validade Externa**, pode-se mencionar a quantidade limitada de Avaliadores. O estudo limitou-se à participação de quatro profissionais especializados em teste de software. Esta restrição pode comprometer a generalização dos resultados, uma vez que não reflete uma gama diversificada de perspectivas que outros profissionais da mesma área poderiam oferecer. Além disso, a eficácia da matriz não foi avaliada em contextos industriais reais. Deste modo, não há garantias de que as prevenções apontadas no catálogo serão eficientes para as respectivas dívidas de testes em todos os cenários.

7 CONCLUSÃO

O presente estudo propõe uma investigação fundamentada na questão: ***"Como as equipes de teste podem proativamente mitigar o Débito de Teste sem depender exclusivamente de ações corretivas?"*** Isso se justifica pelo fato de que a DT pode representar um desafio a longo prazo, e a fase de Teste do ciclo de desenvolvimento de software pode acumular suas próprias Dívidas, tornando assim crucial a prevenção desse cenário.

Desta forma, este estudo buscou elaborar um catálogo abrangente de estratégias para prevenir Dívidas Técnicas no Teste de Software, embasado em uma revisão bibliográfica, com o propósito de que este catálogo pudesse servir como um apoio para equipes de Teste de Software na prevenção de Dívidas de Teste. Para isso, foram estabelecidos relacionamentos entre as estratégias de prevenção de DTs e Dívidas de Teste específicas, evidenciando a potencial capacidade de cada estratégia em prevenir determinado Débito de Teste. A proposta do Catálogo visou oferecer suporte aos profissionais da área tecnológica, especialmente aqueles especializados em Teste de software, para auxiliá-los na tomada de decisões voltadas para a prevenção de Dívidas Técnicas durante a fase de Teste.

Assim, o estudo verificou 98 Dívidas de Teste, as quais foram agrupadas em 17 categorias. Além disso, também elencou-se 47 categorias de estratégias de prevenção de Dívidas Técnicas, estabelecendo suas respectivas relações. Os resultados apontaram ampla concordância entre os avaliadores, os quais possuem experiência prática em contextos nos quais a implementação de ações preventivas foi necessária, corroborando com a proposta apresentada.

Portanto, considerando as respostas obtidas, a proposta emerge como uma ferramenta que pode auxiliar os profissionais na tomada de decisões para prevenir o surgimento de Dívidas Técnicas, oferecendo uma variedade de estratégias para tal.

REFERÊNCIAS

[1] Nathália Accioly, Wylliams Santos, Roberta Fagundes, and Ana Melo. 2023. Gerenciamento de Dívida Técnica no Desenvolvimento Ágil: resultados preliminares de um survey. *Anais do Computer on the Beach* 14 (2023), 442–443.

[2] Danyllo Albuquerque, Everton Guimaraes, Graziela Tonin, Mirko Perkusich, Hyggo Almeida, and Angelo Perkusich. 2022. Comprehending the use of intelligent techniques to support technical debt management. In *Proceedings of the International Conference on Technical Debt*. 21–30.

[3] Danyllo Albuquerque, Everton Guimarães, Graziela Tonin, Pilar Rodriguez, Mirko Perkusich, Hyggo Almeida, Angelo Perkusich, and Ferdinandy Chagas. 2022. Managing Technical Debt Using Intelligent Techniques-A Systematic Mapping Study. *IEEE Transactions on Software Engineering* (2022).

[4] Danyllo Albuquerque, Everton Tavares Guimaraes, Graziela Simone Tonin, Mirko Barbosa Perkusich, Hyggo Almeida, and Angelo Perkusich. 2022. Perceptions of Technical Debt and Its Management Activities - A Survey of Software Practitioners. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering* (, Virtual Event, Brazil,) *(SBES '22)*. Association for Computing Machinery, New York, NY, USA, 220–229. https://doi.org/10.1145/3555228.3555237

[5] Cecilia Apa, Helvio Jeronimo, Luciana M. Nascimento, Diego Vallespir, and Guilherme Horta Travassos. 2020. *The Perception and Management of Technical Debt in Software Startups.* Springer International Publishing, Cham, 61–78. https://doi.org/10.1007/978-3-030-35983-6_4

[6] Cecilia Apa, Martin Solari, Diego Vallespir, and Guilherme Horta Travassos. 2020. A Taste of the Software Industry Perception of Technical Debt and its Management in Uruguay: A survey in software industry. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–9.

[7] Bruno S Aragão, Rossana MC Andrade, Ismayle S Santos, Rute NS Castro, Valéria Lelli, and Ticianne GR Darin. 2019. Testdcat: Catalog of Test Debt subtypes and management activities. In *Testing Software and Systems: 31st IFIP WG 6.1 International Conference, ICTSS 2019, Paris, France, October 15–17, 2019, Proceedings 31*. Springer, 279–295.

[8] Bruno Sabóia Aragão, Rute Nogueira Silveira de Castro, Ismayle de Sousa Santos, Valéria Lelli, and Rossana MC Andrade. 2019. Test Debts identification in a test factory. In *Proceedings of the XVIII Brazilian Symposium on Software Quality*. 296–305.

[9] Bruno Sabóia Aragão. 2019. *TestDCat: Catalog of Test Debt Subtypes and Management Activities.* Dissertação de Mestrado. Universidade Federal do Ceará. http://mdcc.ufc.br/teses-e-dissertacoes/dissertacoes-de-mestrado/cat_view/36-/38-?limit=5&order=name&dir=ASC&start=315

[10] Clara Berenguer, Adriano Borges, Sávio Freire, Nicolli Rios, Robert Ramač, Nebojša Taušan, Boris Pérez, Camilo Castellanos, Darío Correal, Alexia Pacheco, et al. 2023. Investigating the relationship between technical debt management and software development issues. *Journal of Software Engineering Research and Development* (2023), 3–1.

[11] Clara Berenguer, Adriano Borges, Sávio Freire, Nicolli Rios, Nebojsa Tausan, Robert Ramac, Boris Pérez, Camilo Castellanos, Darío Correal, Alexia Pacheco, Gustavo López, Davide Falessi, Carolyn Seaman, Vladimir Mandic, Clemente Izurieta, and Rodrigo Spínola. 2021. Technical Debt is Not Only about Code and We Need to Be Aware about It. In *Proceedings of the XX Brazilian Symposium on Software Quality* (Virtual Event, Brazil) *(SBQS '21)*. Association for Computing Machinery, New York, NY, USA, Article 14, 12 pages. https://doi.org/10.1145/3493244.3493285

[12] Fandi Bi, Birgit Vogel-Heuser, Ziyi Huang, and Felix Ocker. 2023. Characteristics, causes, and consequences of technical debt in the automation domain. *Journal of Systems and Software* (2023), 111725.

[13] Nanette Brown, Yuanfang Cai, Yuepu Guo, Rick Kazman, Miryung Kim, Philippe Kruchten, Erin Lim, Alan MacCormack, Robert Nord, Ipek Ozkaya, et al. 2010. Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*. 47–52.

[14] Zadia Codabux, Melina Vidoni, and Fatemeh H Fard. 2021. Technical debt in the peer-review documentation of r packages: A rOpenSci case study. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 195–206.

[15] Diego Costa, Mariela Cortés, and Eliakim Gama. 2021. On the Relation between Technical Debt Indicators and Quality Criteria in Stack Overflow Discussions. In *Proceedings of the XXXV Brazilian Symposium on Software Engineering* (Joinville, Brazil) *(SBES '21)*. Association for Computing Machinery, New York, NY, USA, 432–441. https://doi.org/10.1145/3474624.3474648

[16] Ward Cunningham. 1992. The WyCash portfolio management system. *ACM Sigplan Oops Messenger* 4, 2 (1992), 29–30.

[17] Luiz Carlos da Fonseca Lage. 2020. Technical Debt of Usability in Software Projects: A Multi-Case Study. *ISYS - BRAZILIAN JOURNAL OF INFORMATION SYSTEMS* (2020).

[18] Dipta Das, Abdullah Al Maruf, Rofiqul Islam, Noah Lambaria, Samuel Kim, Amr S Abdelfattah, Tomas Cerny, Karel Frajtak, Miroslav Bures, and Pavel Tisnovsky. 2022. Technical debt resulting from architectural degradation and code smells: a systematic mapping study. *ACM SIGAPP Applied Computing Review* 21, 4 (2022), 20–36.

[19] Ai Deng. 2020. *Mining technical debt in commit messages and commit linked issues.* Ph. D. Dissertation. Ph. D. dissertation, Faculty of Engineering and Science, University of Groningen.

[20] Arilo Claudio Dias-Neto, R Spinola, and Guilherme Horta Travassos. 2010. Developing software technologies through experimentation: experiences from the battlefield. In *XIII Ibero-American conference on software engineering*.

[21] Sávio Freire, Nicolli Rios, Boris Gutierrez, Darío Torres, Manoel Mendonça, Clemente Izurieta, Carolyn Seaman, and Rodrigo O. Spínola. 2020. Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for Not Paying off Debt Items. In *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering* (Trondheim, Norway) *(EASE '20)*. Association for Computing Machinery, New York, NY, USA, 210–219. https://doi.org/10.1145/3383219.3383241

[22] Sávio Freire, Nicolli Rios, Manoel Mendonça, Davide Falessi, Carolyn Seaman, Clemente Izurieta, and Rodrigo O Spínola. 2020. Actions and impediments for technical debt prevention: results from a global family of industrial surveys. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 1548–1555.

[23] Sávio Freire, Nicolli Rios, Boris Pérez, Camilo Castellanos, Darío Correal, Robert Ramač, Vladimir Mandić, Nebojša Taušan, Gustavo López, Alexia Pacheco, Manoel Mendonça, Davide Falessi, Clemente Izurieta, Carolyn Seaman, and Rodrigo Spínola. 2023. Software Practitioners' Point of View on Technical Debt Payment. *J. Syst. Softw.* 196, C (feb 2023), 40 pages. https://doi.org/10.1016/j.jss.2022.111554

[24] Sávio Freire, Nicolli Rios, Boris Pérez, Camilo Castellanos, Darío Correal, Robert Ramač, Vladimir Mandić, Nebojša Taušan, Alexia Pacheco, Gustavo López, Manoel Mendonça, Clemente Izurieta, Davide Falessi, Carolyn Seaman, and Rodrigo Spínola. 2021. Pitfalls and Solutions for Technical Debt Management in Agile Software Projects. *IEEE Software* 38, 6 (2021), 42–49. https://doi.org/10.1109/MS.2021.3101990

[25] Eliakim Gama, Sávio Freire, Manoel Mendonça, Rodrigo O Spínola, Matheus Paixao, and Mariela I Cortés. 2020. Using stack overflow to assess technical debt identification on software projects. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. 730–739.

[26] Cristian Giannetti. 2023. self-admitted technical debt detection and management in issue tracker systems. (2023).

[27] Tim Klinger, Peri Tarr, Patrick Wagstrom, and Clay Williams. 2011. An enterprise perspective on technical debt. In *Proceedings of the 2nd Workshop on managing technical debt*. 35–38.

[28] Yikun Li, Mohamed Soliman, and Paris Avgeriou. 2020. Identification and remediation of self-admitted technical debt in issue trackers. In *2020 46th Euromicro conference on software engineering and advanced applications (SEAA)*. IEEE, 495–503.

[29] Zengyang Li, Paris Avgeriou, and Peng Liang. 2015. A systematic mapping study on technical debt and its management. *Journal of Systems and Software* 101 (2015), 193–220.

[30] Ana Melo, Roberta Fagundes, Valentina Lenarduzzi, and Wylliams Barbosa Santos. 2022. Identification and measurement of Requirements Technical Debt in software development: A systematic literature review. *Journal of Systems and Software* 194 (2022), 111483.

[31] Glenford J Myers, Tom Badgett, Todd M Thomas, and Corey Sandler. 2004. *The art of software testing*. Vol. 2. Wiley Online Library.

[32] Boris Pérez, Camilo Castellanos, Darío Correal, Nicolli Rios, Sávio Freire, Rodrigo Spínola, Carolyn Seaman, and Clemente Izurieta. 2021. Technical debt payment and prevention through the lenses of software architects. *Information and Software Technology* 140 (2021), 106692.

[33] Boris Pérez, Camilo Castellanos, Darío Correal, Nicolli Rios, Sávio Freire, Rodrigo Spínola, Carolyn Seaman, and Clemente Izurieta. 2021. Technical Debt Payment and Prevention through the Lenses of Software Architects. *Inf. Softw. Technol.* 140, C (dec 2021), 16 pages. https://doi.org/10.1016/j.infsof.2021.106692

[34] Aline Pacheco Primão, Patric da Silva Ribeiro, and Diego Luís Kreutz. 2010. Estudo de Caso: Técnicas de Teste como parte do Ciclo de Desenvolvimento de Software. *Universidade Federal do Pampa-UNIPAMPA, Alegrete, RS* (2010).

[35] Kalle Rindell, Karin Bernsmed, and Martin Gilje Jaatun. 2019. Managing security in software: Or: How i learned to stop worrying and manage the security technical debt. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 1–8.

[36] Nicolli Rios, Rodrigo Oliveira Spinola, Manoel G. de Mendonça Neto, and Carolyn Seaman. 2018. A Study of Factors that Lead Development Teams to Incur Technical Debt in Software Projects. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 429–436. https://doi.org/10.1109/SEAA.2018.00076

[37] Nicolli Rios, Rodrigo Spínola, and Guilherme Travassos. 2022. Exploring Technical Debt on IoT Software Projects. In *Proceedings of the XXI Brazilian Symposium on Software Quality*. 1–10.

[38] Verusca Rocha, Sávio Freire, Manoel Mendonça, and Rodrigo Spínola. 2022. Evaluating a Conceptual Framework for Supporting Technical Debt Management in Testing Activities-A Feasibility Study. In *Proceedings of the 7th Brazilian Symposium on Systematic and Automated Software Testing*. 69–78.

[39] S Pressman Roger and R Maxin Bruce. 2015. *Software engineering: a practitioner's approach.* McGraw-Hill Education.

[40] Hina Saeeda, Muhammad Ovais Ahmad, and Tomas Gustavsson. 2023. Challenges in Large-Scale Agile Software Development Projects. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. 1030–1037.

[41] Ganesh Samarthyam, Mahesh Muralidharan, and Raghu Kalyan Anna. 2017. Understanding test debt. *Trends in software testing* (2017), 1–17.

[42] Jie Tan. 2021. Technical Debt Repayment in Practice. (2021).

[43] Jie Tan, Daniel Feitosa, and Paris Avgeriou. 2020. An empirical study on self-fixed technical debt. In *Proceedings of the 3rd International Conference on Technical Debt*. 11–20.

[44] Jie Tan, Daniel Feitosa, and Paris Avgeriou. 2023. The lifecycle of Technical Debt that manifests in both source code and issue trackers. *Information and Software Technology* 159 (2023), 107216.
[45] Melina C Vidoni. 2021. Software Engineering and R Programming: A Call for Research. *R J.* 13, 2 (2021), 600.
[46] Xin Xia, Zhiyuan Wan, Pavneet Singh Kochhar, and David Lo. 2019. How practitioners perceive coding proficiency. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 924–935.
[47] Xin Xia, Zhiyuan Wan, Pavneet Singh Kochhar, and David Lo. 2019. How practitioners perceive coding proficiency. , 924–935. pages.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009