UNIVERSIDADE FEDERAL DO AMAZONAS – UFAM
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA – ICET
CAMPUS UNIVERSITÁRIO MOYSÉS BENARRÓS ISRAEL
PROJETO PARA EDUCAÇÃO E PESQUISA - SUPER

RELATÓRIO TÉCNICO:
Dívida Técnica em Projetos de Software: Uma Revisão Rápida

Alunos:

Lucas de Souza Pinheiro
lucas.pinheiro@ufam.edu.br

Maryse da Silva Pires
marysepires12@ufam.edu.br

Orientadora:

Prof. Dra. Odette Mestrinho Passos

ITACOATIARA-AM
2024

Sumário

1. INTRODUÇÃO

Segundo Alexandre Bartié (2002), a qualidade de software é um processo sistemático que visa garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos em todas as etapas e artefatos produzidos. No entanto, em projetos de software, quando prazos ou recursos se tornam insuficientes, as organizações tendem a comprometer práticas que buscam garantir a qualidade (Pressman, 2011). De acordo com Rios (2018), uma das consequências dessas más práticas no desenvolvimento de software é o surgimento da Dívida Técnica (DT), ocasionada, geralmente, quando tarefas não são realizadas adequadamente.

O termo conhecido como DT foi introduzido por Ward Cunningham há mais de duas décadas (Cunningham, 1992). Segundo Zazworka (2013), a DT se refere ao equilíbrio entre objetivos de curto e longo prazo no desenvolvimento de software, que surge quando atividades de desenvolvimento são adiadas em prol de resultados imediatos, mas com a possibilidade de trazer consequências negativas a longo prazo. Decisões inadequadas no desenvolvimento de software podem se acumular e levar à formação de uma DT, ocasionando atrasos e retrabalho consideráveis no decorrer do projeto.

Kruchten, Nord e Ozkaya (2012) afirmam que a metáfora da DT está cada vez mais presente na comunidade de desenvolvimento de software. Isso ocorre porque essa metáfora permite que questões fundamentais relacionadas à qualidade, valor e custo do produto sejam compreendidas e comunicadas de maneira mais clara e eficiente (Kruchten *et al.*, 2012). Esse conceito permite que os desenvolvedores considerem não apenas as necessidades imediatas do projeto, mas também os impactos de longo prazo das decisões tomadas durante o processo de desenvolvimento e de que maneira essas decisões comprometem a qualidade final do produto.

Um fator básico para a garantia e manutenção da qualidade do software, é o gerenciamento eficaz da DT, se não for gerenciada, essa dívida cria problemas significativos de longo prazo, como aumento dos custos de manutenção (Brown, *et al.*, 2010). O gerenciamento da DT se dá através de planejamento, monitoramento e controle, sendo um processo contínuo incorporado às atividades de desenvolvimento de software. De acordo com Alves *et al.* (2016) com o objetivo de garantir a produtividade a curto prazo e monitorar o progresso do projeto para evitar que a DT acumulada atrapalhe o desenvolvimento, começaram a ser desenvolvidas técnicas que visam o gerenciamento de DT.

Neste contexto, a atual pesquisa visa identificar os tipos de DT mais comuns e quais as causas, bem como fornecer uma visão geral do efeito dessa problemática na área de desenvolvimento de software. O resultado esperado deste trabalho será uma contribuição para a comunidade de desenvolvimento de software, fornecendo informações importantes para o gerenciamento de DT em projetos de software.

A metodologia de pesquisa, para coletar as informações da literatura, foi baseada em uma Revisão Rápida (RR), do inglês, *Rapid Review*, que são estudos secundários adaptados de revisões sistemáticas convencionais, ajustados para atender às limitações de profissionais, como tempo e custos, visando uma aplicação mais prática na engenharia de software (Cartaxo, Pinto e Soares, 2018). A RRs representa uma síntese ágil do conhecimento, simplificando os elementos do processo de revisão sistemática para fornecer informações de forma oportuna (Khangura, 2012). A condução de uma RR envolve três fases principais: Planejamento,

Execução e Resultado. Cada fase compreende várias etapas específicas, que visam fornecer evidências científicas de forma mais rápida para apoiar a tomada de decisão (Cartaxo, Pinto e Soares, 2018).

A pesquisa foi conduzida pelos alunos que participam do projeto SUPER, no âmbito do Instituto de Ciências Exatas e Tecnologia, que visa estimular a capacitação e a pesquisa em cursos de graduação da Universidade Federal do Amazonas (UFAM).

O restante do artigo está organizado da seguinte maneira. A Seção 2 apresenta alguns conceitos relacionados. A Seção 3 apresenta o método de pesquisa utilizado enquanto a Seção 4 mostra os resultados e as discussões. A Seção 5 apresenta as considerações finais do trabalho, limitações e trabalhos futuros.

2. DÍVIDA TÉCNICA

DT é uma metáfora para artefatos imaturos, incompletos ou inadequados no ciclo de vida de desenvolvimento de software. Dessa forma, a DT é o custo do trabalho adicional causado pela escolha de uma solução fácil em determinado momento, em vez de se utilizar uma abordagem melhor que levaria mais tempo (Martin, 2008). Essas escolhas podem oferecer aumento da produtividade a curto prazo, mas a longo prazo tornam o processo de desenvolvimento de software mais difícil, caro e imprevisível (Guo, 2016).

Segundo Cunningham (1992), a princípio a DT estava focada apenas em atividades que envolviam a codificação, mas com o aumento de pesquisas na área Klinger (2011) enfatiza que a DT não está relacionada exclusivamente ao código. O conceito de DT abrange várias fases do processo de desenvolvimento de software. Alves *et al.* (2016) mencionam que os tipos de DT podem ser classificados em diversas categorias, incluindo infraestrutura, projeto, arquitetura, documentação, teste, código, defeito, requisitos, pessoas, automação de testes, entre outras.

McConnell (2008) classificou a ocorrência de DT em dois tipos: a intencional e a não intencional. A primeira diz respeito aos profissionais tomarem decisões para alcançar resultados de curto prazo, priorizando a entrega ao invés da qualidade, e a não intencional ocorre de forma involuntária e não estrategicamente, sendo muitas vezes causada por atividades mal planejadas devido a profissionais inexperientes ou mudanças no ambiente. Outra abordagem de classificação é de acordo com Fowler (2014), que classifica a DT em Imprudente/Proposital, Prudente/Proposital, Imprudente/Negligente e Prudente/Negligente. Que ainda segundo Fowler (2014), são definidas como:

- Imprudente/Proposital: a equipe de desenvolvimento opta por soluções rápidas, sem muita preocupação com a qualidade.
- Prudente/Proposital: envolve a equipe reconhecendo as limitações e entregando o produto agora, consciente das implicações futuras.
- Imprudente/Negligente: a equipe pode não ter consciência dos princípios básicos de qualidade de software e não perceber as consequências futuras de suas ações
- Prudente/Negligente: ocorre quando a equipe fornece uma solução que agrega valor ao negócio, mas, posteriormente, percebe que a abordagem de design poderia ter sido mais bem fundamentada

Li, Avgeriou e Liang (2015), realizaram um mapeamento para classificar em que parte podem vir as DTs (Dívidas Técnicas) dentro do ciclo de vida de um software, sendo os seus tipos de: requisitos, arquitetura, projeto, código, teste, documentação, infraestrutura, controle, defeito e de construção.

- DT de Código: representa o código mal escrito que viola as melhores práticas de codificação ou regras de codificação (Li, Avgeriou e Liang, 2015)
- DT de Requisitos: refere-se à distância entre a especificação de requisitos ideal e a aplicação efetiva do sistema, com base em pressupostos de domínio e suas restrições (Ernst, 2012).
- DT Arquitetural: a falta de qualidade em alguns aspectos internos, como a manutenção, é resultado de decisões de arquitetura que envolvem concessões (Sousa, 2016).
- DT de Projeto: refere-se a elementos que podem ser revelados por meio da análise do código-fonte, assim como à detecção de violações das melhores práticas de programação orientada a objetos (Guo e Seaman, 2011).
- DT de Teste: refere-se a problemas encontrados nas atividades relacionadas a teste (Porto, 2018).
- DT de Construção: refere-se a deficiências em um sistema de software durante o processo de construção, seja em seu sistema de construção ou no próprio processo, que tornam a compilação um problema (Alves, 2016).
- DT de Documentação: refere-se a problemas na documentação de um software, como documentação inadequada ou insuficiente para o desenvolvimento do projeto (Porto, 2018).
- DT de Infraestrutura: refere-se a problemáticas que se recorrentes na organização de um software, podem atrasar seu desenvolvimento (Alves, 2016).
- DT de Controle de Versão: diz respeito aos problemas no versionamento do código fonte (Zazworka *et al*. 2013).
- DT de Defeito: representa defeitos, erros ou falhas identificados em sistemas de software (Filho, Almeida e Lenço, 2022).

3. PLANEJAMENTO DA RR

3.1. Objetivo e Questões de Pesquisa

O objetivo desta RR é detectar as causas da DT na literatura, coletar informações acerca dos tipos mais comuns e quais seus efeitos em projetos de softwares. Sendo assim, esta RR irá buscar respostas para as seguintes questões de pesquisa (QP):

- **Primeira Questão de Pesquisa (QP1):** Quais são os tipos de DT mais comuns?
- **Segunda Questão de Pesquisa (QP2):** Quais as causas de DT?
- **Terceira Questão de Pesquisa (QP3):** Quais os efeitos da DT em projetos de software?

3.2. Fontes, Idioma e Expressão de Busca

Inicialmente, foram verificados os simpósios apoiados pela Sociedade Brasileira de Computação (SBC), conforme apresentados na Tabela 1, para identificar aqueles que abordavam tópicos de interesse relacionados à DT.

Tabela 1 – Lista dos Simpósios verificados

Simpósios
BRACIS: Brazilian Conference on Intelligent Systems
Brazilian Symposium on Bioinformatics (BSB)
Simpósio Brasileiro de Testes de Software Sistemático e Automatizado
Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software,
Simpósio Brasileiro de Engenharia de Software (SBES)
Simpósio Brasileiro de Linguagens de Programação
Congresso Ibero-Americano em Engenharia de Software (CIBSE)
Congresso da Sociedade Brasileira de Computação (CSBC)
Congresso sobre Tecnologias na Educação (Ctrl+e)
Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais (IHC)
Symposium on Knowledge Discovery, Mining and Learning (KDMiLe)
Latin-American Symposium on Dependable Computing
Congresso Latino-Americano de Software Livre e Tecnologias Abertas (Latinoware)
International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)
Simpósio Brasileiro de Bancos de Dados (SBBD)
Simpósio Brasileiro de Computação Aplicada à Saúde (SBCAS)
Symposium on Integrated Circuits and Systems Design (SBCCI)
Simpósio Brasileiro de Engenharia de Software (SBES)
Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP)
Simpósio Brasileiro de Computação Musical (SBCM)
Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC)
Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)
Congresso Brasileiro de Agroinformática
Simpósio Brasileiro de Informática na Educação (SBIE)
Simpósio Brasileiro de Linguagens de Programação (SBLP)
Simpósio Brasileiro de Métodos Formais (SBMF)
Simpósio Brasileiro de Qualidade de Software (SBQS)
Simpósio Brasileiro de Robótica e Simpósio Latino-Americano de Robótica (SBR/LARS)
Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRS)
Simpósio Brasileiro de Sistemas Colaborativos (SBSC)
Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)
Simpósio Brasileiro de Sistemas de Informação (SBSI)
Conference on Graphics, Patterns and Images (SIBGRAPI)
Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana (STIL)
Simpósio de Realidade Virtual e Aumentada (SVR)
Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia)
Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD)

Fonte: Os Autores (2024)

Dentre os simpósios analisados foi visto que dois deles tinham como tópico de interesse “Dívida Técnica” ou “Gestão de Dívida Técnica”: CibSE e SBES. Os simpósios SBQS, SBSI

e SAST não obtinham o termo como tópico de interesse, mas ao fazer uma busca manual, foi possível verificar que haviam artigos publicados. A análise foi feita de 2018 a 2023. A Tabela 2 mostra os simpósios e os artigos selecionados dentro de cada um.

Tabela 2 – Lista com Simpósios Selecionados e Artigos

Simpósio	Informações e Artigos Selecionados
CIBSE	O tema em questão está presente nos tópicos de interesse, mas não há publicações relacionadas a ele nos anos de 2022 e 2023, sendo esses os anos disponíveis na plataforma SOL
SAST	O tema em questão não está presente nos tópicos de interesse, mas existe publicação com a palavra-chave no título: ● Evaluating a Conceptual Framework for Supporting Technical Debt Management in Testing Activities - A Feasibility Study (2022)
SBES	O tema em questão está presente nos tópicos de interesse. As publicação com a palavra-chave no título, são: ● Perceptions of Technical Debt and Its Management Activities - A Survey of Software Practitioners (2022) ● PriorTD: A Method for Prioritization Technical Debt (2022) ● Towards a Process to Manage Usability Technical Debts (2022) ● Towards an Understanding of Technical Debt in Reference Architectures (2022) ● On the relation between technical debt indicators and quality criteria in Stack Overflow discussions (2021) ● A Decision Support System for Managing Technical Debt: Towards a Systemic Perspective (2021) ● Using Stack Overflow to Assess Technical Debt Identification on Software Projects (2020) ● Anticipating Identification of Technical Debt Items in Model-Driven Software Projects (2020) ● Using Surveys to Build-up Empirical Evidence on Test-Related Technical Debt (2020) ● Documentation Technical Debt: A Qualitative Study in a Software Development Organization (2019) ● Influence of Model Refactoring on Code Debt: A Replicated Study (2019)
SBQS	O tema em questão não está presente nos tópicos de interesse, mas existe publicação com a palavra-chave no título: ● Technical Debt in Brazilian Software Startups: Perceptions of Professionals in Paraná (2023) ● Variability Debt: A Multi-Method Study (2023) ● Exploring Technical Debt on IoT Software Projects (2022) ● Investigating how Agile Software Practitioners Repay Technical Debt in Software Projects (2022) ● Technical Debt on Agile Projects: Managers point of view at Stack Exchange (2022) ● Technical Debt is not Only about Code and We Need to be Aware about It (2021) ● Variability Debt: Characterization, Causes and Consequences (2021) ● Technical Debt Guild: When experience and engagement improve Technical Debt Management (2021) ● Organizing a Set of Empirical Findings on the Causes and Effects of Technical Debt through a Globally Distributed Family of Surveys: Extended Abstract (2021) ● Effects of Visualizing Technical Debts on a Software Maintenance Project (2019) ● Test debts identification in a test factory (2019) ● Technical Debt's State of Practice on Stack Overflow: a Preliminary Study (2019) ● Technologies to Support the Technical Debt Management in Software Projects: A Qualitative Research (2019)

	● A Study on Identification of Documentation and Requirement Technical Debt through Code Comment Analysis (2018) ● Technical Debt Management in Brazilian Software Organizations: A Need, an Expectation or a Fact? (2018)
SBSI	O tema em questão não está presente nos tópicos de interesse, mas existe publicação com a palavra-chave no título: ● Identifying Technical Debt through a Code Comment Mining Tool (2019) ● Identifying and Measuring Technical Debt in Software Requirements: A Supporting Guide (2023)

Fonte: Os Autores (2024)

Após a análise dos tópicos de interesse, foram selecionados os seguintes simpósios como locais de busca para obtenção de publicações relevantes: CIBSE, SBES, SBQS, SBSI e SAST. Essa seleção foi realizada com o objetivo de abranger uma ampla gama de estudos com o tema DT e garantir a obtenção de informações atualizadas e pertinentes. Optou-se por incluir estudos no idioma inglês e português. No entanto, os artigos desejados estavam restritos à plataforma ACM, tornando seu acesso inviável por se tratar de uma base de dados paga. Diante disso, decidiu-se por utilizar o Google Acadêmico para uma nova busca, mantendo os idiomas inglês e português. A seleção das publicações foi realizada em três fases: (i) Busca inicial por publicações na fonte definida, (ii) Primeiro Filtro de Seleção: por meio da análise do título e resumo que continham as palavras-chaves "Dívida Técnica" ou "Technical Debt" e (iii) Segundo Filtro de Seleção: por meio da leitura na íntegra das publicações e que continham, pelo menos, a resposta de uma das QP.

4. CONDUÇÃO E RESULTADOS DA RR

A partir dos critérios seleção, foi possível chegar ao total de 33 artigos selecionados para a extração de dados. Os artigos selecionados, em ordem alfabética, estão na Tabela 3, enquanto a Figura 1 mostra o quantitativo de publicações por ano.

Tabela 3 – Lista dos artigos selecionados

ID	Artigo	Autor(es)	Ano
P01	A Systematic Literature Review on Technical Debt Prioritization: Strategies, Processes, Factors and tools	Lenarduzzi, V, Besker, T, Taibi, D, Martine, A e Fontana F	2021
P02	Analisando Estratégias de Identificação de Dívidas Técnicas	Oliveira, I; Neto, H e Xavier, L	2020
P03	An Empirical Study on Self-Admitted Technical Debt in Dockerfiles	Azuma, H; Matsumoto, S; Kamei, Y e Kusumoto, S	2022
P04	An Empirical Study on the Co-Occurrence Between Refactoring Actions and Self-Admitted Technical Debt Removal	Iammoarino, M; Zampetti, F; Aversano, L e Penta, M	2021
P05	An Overview and Comparison of Technical Debt Measurement Tools	Avgeriou, P; Taibi, D; Ampatzoglou, A; Fontana, F; Besker, T; Chatzigeorgiou, A; Lenarduzzi, V; Martini, A; Moschou, A; Pigazzini, I; Saarimaki, N; Sas, D; Toledo, S e Tsintzira, A	2021
P06	Architectural Technical Debt in Microservices	Toledo, S; Martini, A; Przybyszewska, A e Sjøberg, D	2019
P07	ATDx: Building an Architectural Technical Debt Index	Verdecchia, R; Lago, P; Malavolta, I e Ozkaya, I.	2020

P08	Automating Change-Level Self-Admitted Technical Debt Determination	Yan, M; Xia, X; Shihab, E; Lo, D; Yin, J e Yang, X	2018
P09	A Tertiary Study on Technical Debt: Types, Management Strategies, Research Trends, and base Information for Practitioners	Rios, N; Neto, M e Spínola, R	2018
P10	Common Causes and Effects of Technical Debt in Serbian IT: InsighTD Survey Replication	Ramač, R; Mandič, V; Taušan, N; Rios, N; Neto, M; Seaman, C e Spínola, R	2020
P11	Conceptualisation of Using Technical Debt 2 to Measure the Innovation Level of New Product – 3 Selected Issues	Filipowicz, P	2021
P12	Embracing Technical Debt, from a Startup Company Perspective	Besker, T; Martini, A; Lokuge, R; Blincoe, K e Bosch, J	2018
P13	Familiarity, Causes and Reactions of Software Practitioners to the Presence of Technical Debt: A Replicated Study in the Chilean Software Industry	Pérez, B; Brito, J; Astudillo, H; Correal, D; Rios, N; Spínola, R; Mendonça, M e Seaman, C	2019
P14	How do Developers Fix Issues and Pay Back Technical Debt in the Apache Ecosystem?	Digkas, G; Lungu, M; Avgeriou, P; Chatzigeorgiou, A e Ampatzoglou, A	2018
P15	Identifying Design and Requirement Self-Admitted Technical Debt using N-gram IDF	Wattanakriengkra, S; Maipradit, R; Hata, H; Choetkiertikul, M; Sunetnanta, T e Matdumoto, K	2018
P16	Increasing Awareness for Potential Technical Debt in the Engineering of Production Systems	Ocker, F; Seitz, M; Oligschläger, M; Zou, M e Heuser, B.	2019
P17	Introducing the Concept of Technical Debt to Smart Grids: A System Engineering Perspective	Schütz, J e Uslar, M.	2019
P18	Investigating the Point of View of Project Management Practitioners on Technical Debt - A Study on Stack Exchange	Gomes, F; Santos, E; Freire, S; Mendonça, M; Mendes, T e Spínola, R	2022
P19	Managing Architectural Technical Debt: A Unified Model and Systematic Literature Review	Besker, T; Martini, A e Bosch, J	2018
P20	Preventing Technical Debt with the TAP Framework for Technical Debt Aware Management	Wiese, M; Rachow, P; Riebisch, M e Schwarze, J	2022
P21	Security Risk Assessment and Management as Technical Debt	Rindell, K e Holvitie J	2019
P22	Technical Debt and Agile Software Development Practices and Processes: An Industry Practitioner Survey	Holvitie, J; Licorish, S; Spínola, R; Hyrybsalmi, S; MacDonell, S; Mendes, T; Buchan, J e Leppänen, V	2018
P23	Technical Debt Aware Estimations in Software Engineering: A Systematic Mapping Study	Klimczyk, P e Madeyski, L	2020
P24	Technical Debt in Hardware Systems and Elements	Rosser, L e Ouzzif, Z	2021
P25	Technical Debt in Large-Scale Distributed Projects: An Industrial Case Study	Sousa, A; Rocha, L; Britto, R; Gong, Z e Lyu F	2021
P26	Technical Debt Management Automation: State of the Art and Future Perspectives	Biazotto, J; Feitosa, D; Avgeriou, P e Nakagawa, E	2023
P27	Technical Debt Prioritization: Taxonomy, Methods Results and Practical Characteristics	Pina, D; Goldaman, A e Tonin, G	2021
P28	Technical Debt Tracking: Current State of practice: A Survey and Multiple Case Study in 15 Large Organizations	Martini, A, Besker, T e Bosch, J	2018
P29	The Lifecycle of Technical Debt that Manifests in Both Source Code and Issue Trackers:	Tan, J; Feitosa, D e Avgeriou, P	2023
P30	The Role of Awareness and Gamification on Technical Debt Management	Crespo. Y; Nozal, C; Sánchez, R; Tasis, M e Piattini, M	2022
P31	Toward a Technical Debt Conceptualization for Serverless Computing	Lenarduzzi, V; Daly, J; Martini, A; Panichella, S e Tamburri	2021

P32	Uma Análise da Relação entre Code Smells e Dívida Técnica Auto-aAdmitida	Gomes, F; Mendes, T; Spínola, R;Mendonça, M e Farias, M	2019
P33	Who (Self) Admits Technical Debt?	Fucci, G; Zampetti, F; Serebrenik, A e Penta, M	2020

Fonte: Os Autores (2024)

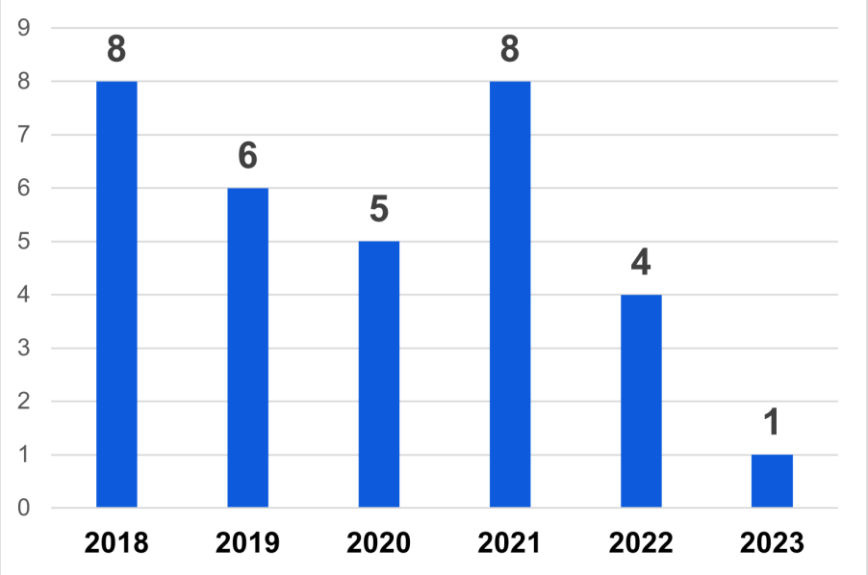

Figura 1 – Quantitativo de publicações por ano

Fonte: Os Autores (2024)

A pesquisa identificou um total de 19 tipos diferentes de DT que impactam projetos de software, conforme detalhado na Tabela 4 e mostrado com quantitativo na Figura 2. Entre esses tipos, a DT de Arquitetura foi a mais citada, com um total de 13 citações. Segundo o estudo [P19], essa forma de dívida pode dificultar a integração de novos recursos, resultando em custos financeiros significativos devido à necessidade de manutenção excessiva da arquitetura. Em projetos que trabalham com hardware, erros na arquitetura, nos requisitos e no projeto também levam a uma taxa de defeitos mais alta no software [P24]. Esse tipo de DT, muitas vezes enfatiza importância de uma gestão diligente da dívida técnica desde as fases iniciais do desenvolvimento, visando mitigar riscos e assegurar a qualidade e a sustentabilidade dos projetos de software ao longo do tempo.

Tabela 4 – Tipos de DT identificadas nos artigos

ID	Tipo de DT	Publicações
DT01	Dívida de Arquitetura	[P01], [P05], [P06], [P07], [P09] e [P12] [P13], [P16], [P19], [P23], [P28], [P29] e [P31]
DT02	Dívida de Código	[P01], [P05], [P09], [P12], [P14], [P18], [P28], [P29] [P30] e [P31]
DT03	Dívida Técnica Auto-Admitida	[P02], [P03], [P04], [P08], [P15], [P26], [P32] e [P33]
DT04	Dívida de Design	[P01], [P05], [P09], [P12], [P16] e [P30]
DT05	Dívida de Teste	[P09], [P18], [P28], [P29], [P30] e [P31]
DT06	Dívida de Documentação	[P09], [P16], [P18], [P28] e [P30]
DT07	Dívida de Defeito	[P09], [P18] e [P29]
DT08	Dívida de Requisitos	[P09], [P16] e [P18]
DT09	Dívida de Build	[P09] e [P18]
DT10	Dívida de Processo	[P09] e [P18]

DT11	Dívida de Usabilidade	[P09], [P18]
DT12	Dívida de Pessoas	[P09] e [P18]
DT13	Dívida de Requerimento	[P18] e [P28]
DT14	Dívida de Projeto	[P18] e [P29]
DT15	Dívida de Versionamento	[P18]
DT16	Dívida de Segurança	[P21]
DT17	Dívida Social	[P28]
DT18	Dívida de Hardware	[P24]
DT19	Dívida de Infraestrutura	[P28]

Fonte: Os Autores (2024)

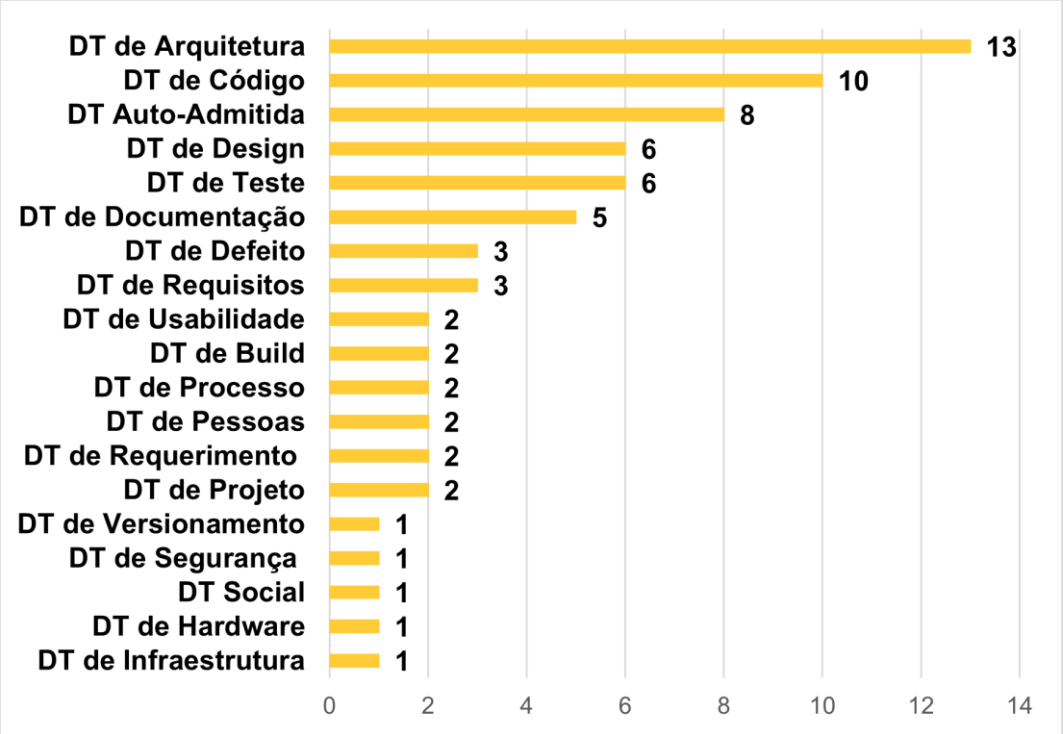

Figura 2 – DT citadas nos artigos

Fonte: Os Autores (2023)

De acordo com o estudo de Tan, Feitosa e Avgeriou, (2023), a DT de Código [DT02] pode ter diversos indicadores, sendo eles:

- **Complexidade do Código:** Identifica trechos de código com complexidade acidental, exigindo ações adicionais de refatoração para reduzir essa complexidade e melhorar a manutenibilidade.
- **Código Morto:** Aponta partes do código que não são mais utilizadas e precisam ser removidas para reduzir a complexidade do sistema e melhorar a legibilidade.
- **Duplicação de Código:** Indica a presença de trechos de código que se repetem, em vez de serem encapsulados em funções reutilizáveis, o que pode aumentar a redundância e dificultar a manutenção.
- **Baixa Qualidade do Código:** Refere-se a trechos de código com qualidade inferior, caracterizados por ilegibilidade, inconsistência ou violações das convenções de codificação, prejudicando a compreensão e a manutenção do código.

- **Incorreção Multithread:** Identifica situações em que o código thread-safe não está corretamente implementado, podendo resultar em problemas de sincronização ou de eficiência em ambientes de múltiplos threads.
- **Algoritmo Lento:** Sinaliza a utilização de algoritmos que operam de forma ineficiente, impactando no desempenho do sistema, sendo necessário buscar alternativas mais eficazes.

Esses indicadores destacam áreas críticas no código que requerem atenção durante o processo de desenvolvimento e manutenção de software, visando garantir a qualidade, eficiência e confiabilidade do sistema.

Outro tipo de DT destacado foi a Dívida Técnica Auto-Admitida (DT03), que recebeu 8 citações. De acordo com Potdar e Shihab (2014), este termo é utilizado para descrever situações em que os desenvolvedores deliberadamente introduzem dívidas técnicas no código e as documentam por meio de comentários. Nas publicações [P02], [P03], [P04], [P08], [P15], [P26], [P32] e [P33], notou-se que a DT03 pode ser de diferentes tipos, sendo eles: de código, design, defeito, documentação, requisitos, teste, projeto, arquitetura e *build*.

A Dívida Técnica (DT04) de Design pode ser identificada através da análise de comentários em código fonte. Segundo Maldonado e Shilab (2018), dívidas de design e requisitos são os tipos mais comuns de DT03 encontradas em comentários de código-fonte. Comentários referentes a DT04 podem incluir violações aos princípios de um bom design orientado a objetos, posicionamento inadequado de código, métodos extensos, falta de abstração, implementações deficientes e soluções de atalho, conforme discutido por Alves et al. (2014).

O estudo de Tan *et al*. (2023) mostra que quando se trata de DT de Teste, pode-se citar algumas causas, destaca-se o custo dos testes, frequentemente elevado e pouco atrativo, bem como a presença de testes instáveis, caracterizados por falhas recorrentes. Outro aspecto relevante é a ausência de testes, como exemplificado pela introdução de novas funcionalidades na aplicação sem a correspondente criação de testes relacionados a essas funcionalidades. Além disso, a baixa cobertura de testes também foi identificada como uma questão significativa, na qual apenas uma parte do código é submetida a testes, deixando outras partes sem a devida validação.

A DT de Documentação aborda as lacunas identificadas na documentação do projeto de software. Esta falha pode manifestar-se de diversas formas, incluindo a ausência de documentação, documentação inadequada, desatualizada ou incompleta [P09]. Esses problemas podem comprometer significativamente o entendimento do projeto, dificultar a manutenção e prejudicar a eficiência do processo de desenvolvimento. Segundo o estudo de Mendes, Cerdeiral e Santos (2019), os participantes dessa pesquisa citaram como umas das causas a documentação desatualizada, onde foi relatado que os problemas de documentação desatualizada poderiam ser mitigados caso os processos fossem respeitados ou até mesmo reestruturado. A DT de Defeito descreve bugs cuja resolução foi adiada por motivos de prioridade ou limitação de teste [P03]. Existe uma demora nessa correção e muitas vezes esses defeitos são ignorados [P29].

Quando se analisa as causas da DT, é evidente a existência de fatores interligados. O estudo conduzido por [P13], por exemplo, investigou uma empresa industrial por meio de pesquisas de opinião, abordando questões não técnicas que contribuem para a DT. Segundo os arquitetos e desenvolvedores de software entrevistados, a escassez de profissionais qualificados emergiu como a causa mais mencionada. Por outro lado, os gerentes de projeto apontaram para questões relacionadas ao planejamento e gerenciamento, destacando a ineficácia na gestão de prazos como uma das principais causas.

Ademais, o estudo [P20] identifica o prazo como uma causa significativa de DT, e o [P10] menciona além do prazo, falhas no gerenciamento de projetos, falta de experiência, ausência de testes adequados, má conduta, priorização da quantidade sobre a qualidade, escassez de profissionais qualificados, falta de adoção de boas práticas, ausência de refatoração e má alocação de recursos como fatores contribuintes para o surgimento do DT. Essa variedade de fatores destaca a complexidade envolvida na gestão e mitigação do DT em ambientes de desenvolvimento de software.

Segundo Klimczyk e Madeyski (2020), a pressão empresarial para a entrega do produto é identificada como a principal causa de problemas de desenvolvimento de DT, destacando também a carência de procedimentos voltados para o gerenciamento desses problemas. Essa preocupação é reiterada no estudo [P20], que ressalta os prazos apertados e a instabilidade dos sistemas como fatores recorrentes de causa de DT. No estudo [P01] cita-se uma série de causas, incluindo imprevisibilidade nos negócios ou influências ambientais internas e externas à organização, enquanto em [P22] a arquitetura inadequada e estrutura inadequada foram as causas mais citadas.

No estudo apresentado em [P26], os autores identificaram diversas outras causas de problemas de desenvolvimento de software, corroborando com descobertas anteriores. Entre essas causas, destacam-se más decisões de design, a ausência de uma pessoa chave, negligência em melhorias técnicas, deficiências na educação, dificuldades de comunicação e, como já mencionado anteriormente, problemas relacionados ao planejamento do projeto. Seis fatores organizacionais que exercem influência no acúmulo de débito técnico em startups de software incluem a experiência dos desenvolvedores, o conhecimento em software dos fundadores das startups, o crescimento do quadro de funcionários, a incerteza e a ausência de processo de desenvolvimento e autonomia [Besker *et al*, 2018].

Dentro dessa ánalise, destaca-se a escassez de profissionais qualificados [P13], problemas de gerenciamento de projetos, prazos apertados ([P10] e [P10]), pressão empresarial ([P20] e [P23]), instabilidade dos sistemas [P20], entre outros, como contribuintes para o surgimento do DT. A complexidade envolvida na gestão e mitigação do DT é evidenciada pela diversidade desses fatores, que variam desde aspectos técnicos até questões organizacionais e de planejamento. Além disso, a presença do DT em diferentes contextos, como startups de software, ressalta a importância de considerar fatores específicos do ambiente organizacional ao lidar com esse desafio. Assim, compreender a natureza interconectada e diversificada das causas do DT é fundamental para implementar estratégias eficazes de prevenção e resolução desse problema no desenvolvimento de software.

Em relação aos efeitos da DT, pode-se verificar que na unidade de TI estudado em [P20], a acumulação de DT provocou um aumento significativo nos tempos de ciclo, resultando em erros desnecessários nos sistemas e gerando insatisfação entre os desenvolvedores. Entretando, a gestão de TI não tinha conhecimento prévio da existência desses problemas, uma vez que não estava ciente de ter contraído qualquer tipo de DT. Essa falta de consciência ressalta a importância de uma vigilância contínua e de um monitoramento proativo da DT, a fim de evitar impactos adversos não previstos.

Em um estudo de caso conduzido por pesquisadores da [P07], focado na análise da DTA em microsserviços, foi investigado o impacto financeiro e operacional dessas dívidas sobre os microserviços. O estudo destacou o alto custo associado à manutenção, bem como o acoplamento excessivo entre os serviços, resultando em maior esforço para atividades como engenharia reversa de software, especialmente quando a documentação ou o código-fonte estavam ausentes. O custo adicional em manutenção também foi citado em [P27], onde os resultados ressaltam a importância de uma gestão proativa da DT para evitar consequências adversas na eficiência operacional e na qualidade do software.

Com base na pesquisa realizada no estudo [P10], foram identificados 10 efeitos da DT, sendo os mais proeminentes: baixa manutenibilidade, aumento do esforço necessário para lidar com o DT acumulado (o que pode ser interpretado como uma perda de tempo do ponto de vista da gestão), retrabalho (que é uma necessidade de revisar e reescrever partes do código devido a falhas ou deficiências relacionadas ao DT), baixa qualidade externa (que pode apresentar problemas visíveis para os usuários finais, como erros frequentes) e aumento de custos tanto em manutenção quanto desenvolvimento.

O impacto da DT de teste se manifesta na identificação tardia de defeitos e na elevada densidade de falhas. Defeitos que não são detectados durante as fases iniciais devido a testes limitados, metodologias de teste inadequadas e ambientes de teste não representativos necessitam de correção posteriormente, acarretando custos elevados [P24]. O aumento da DT resulta em uma redução na capacidade de atender às necessidades dos clientes, deixando-os perplexos quanto ao motivo pelo qual mesmo pequenas modificações exigem um longo período de implementação [P11].

Em suma, a análise dos estudos sobre DT destaca a importância crítica de uma gestão proativa e contínua para evitar os efeitos adversos observados em diferentes contextos. A falta de consciência da gestão sobre a existência desses problemas ressalta a necessidade de uma vigilância constante e de um monitoramento proativo da DT. Portanto, é essencial que as organizações implementem estratégias eficazes de gestão de DT para mitigar esses riscos e garantir a entrega de software de alta qualidade que atenda às necessidades dos clientes de forma eficiente e oportuna.

5. CONSIDERAÇÕES FINAIS

A DT, embora seja um termo relativamente novo na área de Software, se refere a práticas antigas que impactam projetos desde a fase inicial até a manutenção do software após sua entrega. Através da RR, foram identificados 19 tipos de DT, com a Dívida Técnica de

Arquitetura se destacando com 13 citações. Esta categoria de DT ganha destaque devido à sua influência abrangente na estrutura e na integridade do sistema, podendo resultar em dificuldades significativas ao longo do ciclo de vida do projeto. Em seguida a Dívida Técnica de Código, com 10 menções, este tipo de DT é frequentemente observado devido a práticas de codificação apressadas, falta de padronização, e complexidade desnecessária, o que pode prejudicar a legibilidade, manutenibilidade e desempenho do código-fonte. A terceira categoria mais mencionada foi a dívida Auto-Admitida, com 8 citações, que diz respeito às falhas reconhecidas e aceitas pelos desenvolvedores durante o processo de desenvolvimento.

Quando se aborda as causas da DT, foi visto que ela não está limitada apenas a aspectos técnicos, mas também está intrinsecamente relacionada a fatores organizacionais, de planejamento e de gestão de projetos. A qualidade do software é diretamente influenciada pela eficácia da gestão de projetos, uma vez que a presença de falhas nessa gestão pode resultar em deficiências no desenvolvimento do software. A escassez de profissionais qualificados, prazos apertados, pressões empresariais e falta de consciência da gestão sobre esses problemas são apenas alguns exemplos das muitas causas identificadas que contribuem para o surgimento da DT.

Os efeitos da DT são extensos e multifacetadas, impactando não apenas a manutenibilidade e a qualidade do software, mas também o bem-estar da equipe de desenvolvimento. A necessidade de lidar com a DT quando ela "cobra seus juros" demanda um esforço adicional, gerando frustração, desmotivação e estresse. Além disso, a presença de DT pode levar a um aumento na carga de trabalho devido à necessidade de correções e à dificuldade de implementar mudanças no código existente. Isso, por sua vez, eleva os custos operacionais e impacta diretamente na qualidade do produto, influenciando a satisfação dos clientes.

Diante desse cenário, torna-se crucial uma gestão proativa e contínua da DT para mitigar seus efeitos adversos e garantir a entrega de software de alta qualidade. Isso envolve a implementação de estratégias eficazes de monitoramento, identificação e resolução da DT, bem como a conscientização de todos os envolvidos sobre sua importância e impacto nos projetos de software. Portanto, os resultados da RR destacam a necessidade de uma abordagem abrangente e integrada para lidar com a DT, envolvendo tanto aspectos técnicos quanto organizacionais.

Embora o trabalho tenha sido realizado cuidadosamente, é importante destacar algumas limitações relacionadas ao RR. A ausência dos artigos inicialmente selecionados na plataforma SOL representou uma limitação significativa para a RR, já que os simpósios escolhidos possuíam relevância extrema e poderiam fornecer informações únicas para enriquecer ainda mais esta pesquisa. Além disso, a busca manual realizada no Google Acadêmico pode não ter sido abrangente o suficiente, esse processo se mostrou cansativo e limitado, possivelmente levando à omissão de artigos importantes na seleção final. Ademais, seria recomendável explorar outras bases de dados que contivessem estudos pertinentes à área de pesquisa, a fim de garantir uma análise mais abrangente e abalizada.

Como trabalhos futuros, recomenda-se considerar algumas melhorias e expansões da pesquisa já realizada. Uma proposta seria estender o RR para além da fonte de busca escolhida, seria importante aplicar também aos simpósios inicialmente selecionados. Além disso, aplicar

novas questões de pesquisas relacionadas ao gerenciamento de DT e realizar uma análise de dados em relação as técnicas de gerenciamento utilizadas, descrevendo as vantagens e desvantagens de sua utilização e em qual contexto é melhor ser aplicado.

AGRADECIMENTOS

Esta pesquisa, realizada no âmbito do Projeto Samsung-UFAM de Ensino e Pesquisa (SUPER), de acordo com o Artigo 39 do Decreto n°10.521/2020, foi financiada pela Samsung Eletrônica da Amazônia Ltda, nos termos da Lei Federal n°8.387/1991, através do convênio 001/2020 firmado com a UFAM e FAEPI, Brasil

REFERÊNCIAS

Alves, N., Mendes, T., Mendonça, M., Spínola, R., Shull, F. e Seaman, C. (2016). **Identification and management of technical debt: a systematic mapping study.** Information and Software Technology, v. 70, p. 100-121.

Alves, N., Ribeiro, L., Caires, V., Mendes, T. e Spínola, R. (2014). **Towards an Ontology of Termos on Technical Debt**. VI Workshop Internacional sobre Gestão de Dívida Técnica, Canadá, 2014, p. 1-7.

Azuma, H., Matsumoto, S., Kamei, Y. e Kusumoto, S. (2022**). An Empirical Study on Self-Admitted Technical Debt in Dockerfiles.** Empirical Software Engineering, v. 27, n. 2, p. 49.

Avgeriou, P.; Taibi, D.; Ampatzoglou, A.; Fontana, F.; Besker, T.; Chatzigeorgiou, A.; Lenarduzzi, V.; Martini, A.; Moschou, A.; Pigazzini, I.; Saarimaki, N.; Sas, D; Toledo, S. e Tsintzira, A. (2020). **An Overview and Comparison of Technical Debt Measurement Tools.** IEEE Software, v. 38, n. 3, p. 61-71.

Bartié, A. (2002). **Garantia da Qualidade de Software: Adquirindo Maturidade Organizacional**. Rio de Janeiro: Elsevier.

Besker, T., Martini, A., Lokuge, R., Blincoe, K. e Bosch, J. (2018). **Embracing Technical Debt, From a Startup Company Perspective.** XXXIV International Conference on Software Maintenance and Evolution, Madri, p. 415-425.

Besker, T., Martini, A., e Bosch, J. (2018). **Managing Architectural Technical Debt: A Unified Model and Systematic Literature Review.** Journal of Systems and Software, v. 135, p. 1-16.

Brown, N.; Cai, Y.; Guo, Y.; Kazman, R.; Kim, M.; Kruchten, P.; Lim, E.; MacCormack, A.; Nord, R.; Ozkaya, I.; Sangwan, R.; Seaman, C.; Sullivan., K. e Zazworka, N. (2019). **Managing Technical Debt in Software-Reliant Systems. Proceedings of the FSE/SDP**. Workshop on Future of Software Engineering Research, p. 47-52.

Biazotto, J., Feitosa, D., Avgeriou, P. e Nakagawa, E. (2023). **Technical Debt Management Automation: State of the Art and Future Perspectives**. Information and Software Technology, v. 167, p. 107375.

Cartaxo, B., Pinto, G. e Soares, S. (2018). **The Role of Rapid Reviews in Supporting Decisionmaking in Software Engineering Practice**. International Conference on Evaluation and Assessment in Software Engineering, p. 24–34.

Cunningham, W. (1992). **The WyCash Portfolio Management System**. ACM SIGPLAN OOPS Messenger, v. 4, n. 2, p. 29-30.

Crespo, Y., Lopez-Nozal, C., Marticorena-Sanchez, R., Gonzalo-Tasis, M. e Piattini, M. (2022). **The Role of Awareness and Gamification on Technical Debt Management**. Information and Software Technology, v. 150, p. 106946.

Digkas, G., Lungu, M., Avgeriou, P., Chatzigeorgiou, A. e Ampatzoglou, A. (2018). **How do Developers Fix Issues and Pay Back Technical Debt in the Apache Ecosystem?**. XXV International Conference on Software Analysis, Evolution and Reengineering, Itália, p. 153-163.

Ernst, N. **On the Role of Requirements in Understanding and Managing Technical Debt**. Third International Workshop on Managing Technical Debt (MTD), p. 61-64, 2012.

Filho, M., Almeida, I. e Lenço, J. (2023). **Aplicação do Portal do Conhecimento e Café do Conhecimento para Aprimorar a Compreensão dos Profissionais de TI sobre Dívida Técnica: Estudos Preliminares.** Anais do Congresso Internacional de Conhecimento e Inovação – Ciki, v. 1, n. 1.

Fowler, M. **Technical Debt Quadrant.** (2009). Disponível em: https://martinfowler.com/bliki/TechnicalDebtQuadrant.html. Acesso em: 17 fev. 2024.

Fucci, G., Zampetti, F., Serebrenik, A., e Di Penta, M. (2020). **Who (Self) Admits Technical Debt?.** XXXVI International Conference on Software Maintenance and Evolution, Australia, p. 672-676.

Gomes, F., Mendes, T., Spínola, R., Mendonça, M. e Farias, M. (2019). **Uma Análise da Relação entre Code Smells e Dívida Técnica Auto-Admitida**. VII Workshop de Visualização, Evolução e Manutenção de Software, Salvador, p. 25-32.

Gomes, F., Santos, E., Freire, S., Mendonça, M., Mendes, T. e Spínola, R. (2022**). Investigating the Point of View of Project Management Practitioners on Technical Debt: A Preliminary Study on Stack Exchange.** XX International Conference on Technical Debt, New York, p. 31–40.

Guo, Y., Spínola, R. e Seaman, C. (2016). **Exploring the Costs of Technical Debt Management – a Case Study**. Empirical Software Engineering, v. 21, n. 1, p. 159-182.

Guo, Y. e Seaman, C. **A Portfolio Approach to Technical Debt Management**. Workshop on Managing Technical Debt, p. 31-34, 2011.

Holvitie, J., Licorish, S., Spínola, R., Hyrynsalmi, S., MacDonell, S., Mendes, T., Buchan, J. e Leppänen, V. (2018). **Technical Debt and Agile Software Development Practices and Processes: An Industry Practitioner Survey.** Information and Software Technology, v. 96, p. 141-160.

Iammarino, M., Zampetti, F., Aversano, L. e Di Penta, M. (2021). **An Empirical Study on the Co-Occurrence Between Refactoring Actions and Self-Admitted Technical Debt Removal**. Journal of Systems and Software, v.178.

Khangura, S.; Konnyu, K.; Cushman, R.; Grimshaw, J. e Moher, D. (2012). **Evidence Summaries: The Evolution of a Rapid Review Approach.** Systematic Reviews, v.1, n. 10.

Klimczyk, P. e Madeyski, L. (2020). **Technical Debt Aware Estimations in Software Engineering: A Systematic Mapping Study.** e-Informatica Software Engineering Journal, v. 14, n. 4.

Klinger, T., Tarr, P., Wagstrom, P. e Williaws, C. (2011**). An Enterprise Perspective on Technical Debt**. Workshop on Managing Technical Debt, p. 35-38.

Klimczyk, P. e Madeyski, L. (2020). **Technical Debt Aware Estimations in Software Engineering: A Systematic Mapping Study**. e-Informatica Software Engineering Journal, v.14, n.1, p. 61–76.

Kruchten, P., Nord, R., Ozkaya, I. (2012**). Technical Debt: From Metaphor to Theory and Practice.** IEEE software, v. 29, n. 6, p. 18-21.

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A. e Fontana, F. (2021). **A Systematic Literature Review on Technical Debt Prioritization: Strategies, Processes, Factors, and Tools**. Journal of Systems and Software, v.171.

Lenarduzzi, V., Daly, J., Martini, A., Panichella, S. e Tamburri, D. (2020). **Toward a Technical Debt Conceptualization for Serverless Computing.** IEEE Software, v. 38, n. 1, p. 40-47.

Li, Z., Avgeriou, P. e Liang, P. (2015). **A Systematic Mapping Study on Technical Debt and its Management. Journal of Systems And Software**, v. 101, p. 193-220.

Martin, R. (2008). **Clean Code: a Handbook of Agile Software Craftsmanship**. New Jersey : Prentice Hall.

Martini, A., Besker, T. e Bosch, J. (2018). **Technical Debt Tracking: Current State of Practice: A Survey and Multiple Case Study in 15 Large Organizations**. Science of Computer Programming, v.163, p.42-61.

Maldonado, E. e Shihab, E. (2015). **Detecting and Quantifying Different Types of Self-Admitted Technical Debt**. VII International Workshop on Managing Technical Debt (MTD0), p. 9-15.

Mendes, L., Cerdeiral, C., e Santos, G. (2019). **Documentation Technical Debt: A Qualitative Study in a Software Development Organization.** Simpósio Brasileiro de Engenharia de Software, Porto Alegre, p-23.

McConnel, S. (2008). **Managing Technical Debt.** Construx Software Builders.

Oliveira, I., Marques, H. E Xavier, L. (2020). **Analisando Estratégias para Identificação de Dívidas Técnicas.** Workshop de Visualização, Evolução e Manutenção de Software, Evento Online, p. 9-16.

Ocker, F., Seitz, M., Oligschläger, M., Zou, M. e Vogel-Heuser, B. (2019). **Increasing Awareness for Potential Technical Debt in the Engineering of Production Systems**. XVII International Conference on Industrial Informatics, Finlândia, p. 478-484.

Paweł, F. (2021). **Conceptualisation of Using Technical Debt to Measure the Innovation Level of New Product–Selected Issues**. Scientific Papers of Silesian University of Technology, v. 153, p. 89.

Pérez, B.; Brito, J.; Astudillo, H.; Correal, D.; Rios, N.; Spínola, R.; Mendonça, M. e Seaman, C. (2019). **Familiarity, Causes and Reactions of Software Practitioners to the Presence of Technical Debt: A Replicated Study in the Chilean Software Industry.** XXX International Conference of the Chilean Computer Science Society, Chile, p. 1-7.

Pressman, R. (2011**). Engenharia de Software: uma Abordagem Profissional.** 7. ed. São Paulo: Pearson Makron Books.

Pina, D., Goldman, A. e Tonin, G. (2021). **Technical Debt Prioritization: Taxonomy, Methods Results, and Practical Characteristics.** XLVII Euromicro Conference on Software Engineering and Advanced Applications, Espanha, p. 206-213.

Potdar, A. e Shihab, E. (2014). **An Exploratory Study on Self-Admitted Technical Debt**. IEEE International Conference on Software Maintenance and Evolution (ICSM), Canadá, p. 91-100.

Ramač, R., Mandić, V., Taušan, N., Rios, N., de Mendonca Neto, M. G., Seaman, C. e Spínola, R. (2020). **Common Causes and Effects of Technical Debt in Serbian it: Insightd Survey Replication.** XLIII Euromicro Conference on Software Engineering and Advanced Applications, Evento Online, p. 354-361.

Rios, N., Mendonça, M., Spínola, R. (2018). **A Tertiary Study on Technical Debt: Types, Management Strategies, Research Trends, and Base Information for Practitioners.** Information and Software Technology, v. 102, p. 117-145.

Rindell, K. e Holvitie, J. (2019**). Security Risk Assessment and Management as Technical Debt**. International Conference on Cyber Security and Protection of Digital Services, Reino Unido, p. 1-8.

Rosser, L. e Ouzzif, Z. (2021). **Technical Debt in Hardware Systems and Elements.** XLII IEEE Aerospace Conference, Montana, p. 1-10.

Sousa, C. (2016). **Mapa de Apoio à Gestão de Dívida Técnica no Processo de Teste de Software**. 110 f. Dissertação (Mestrado) - Curso de Ciência da Computação - Universidade Federal de Pernambuco, Recife.

Sousa, A., Rocha, L., Britto, R., Gong, Z. e Lyu, F. (2021). **Technical Debt in Large-Scale Distributed Projects: An Industrial Case Study.** XXVIII International Conference on Software Analysis, Evolution and Reengineering, Havaí, p. 590-594.

Tan, J., Feitosa, D. e Avgeriou, P. (2023). **The Lifecycle of Technical Debt that Manifests in Both Source Code and Issue Trackers**. Information and Software Technology, v.159.

Toledo, S., Martini, A., Przybyszewska, A. e Sjøberg, D. (2019). **Architectural Technical Debt in Microservices.** International Conference on Technical Debt, Canadá, p. 78-87

Verdecchia, R., Lago, P., Malavolta, I. e Ozkaya, I. (2020). **ATDx: Building an Architectural Technical Debt Index**. XV International Conference on Evaluation of Novel Approaches to Software Engineering, Evento Online, p. 531-539.

Wattanakriengkrai, S., Maipradit, R., Hata, H., Choetkiertikul, M., Sunetnanta, T., e Matsumoto, K. (2018). **Identifying Design and Requirement Self-Admitted Technical Debt Using N-Gram IDF**. IX International Workshop on Empirical Software Engineering in Practice, Japão, p. 7-12.

Wiese, M., Rachw, P., Riebisch, M. e Schwarze J. (2022). **Preventing Technical debt with the TAP framework for Technical Debt Aware Management.** Information and Software Technology, v. 148.

Yan, M., Xia, X., Shihab, E., Lo, D., Yin, J. Yang, X. (2018). **Automating Change-Level Self-Admitted Technical Debt Determination**. IEEE Transactions on Software Engineering, v. 45, n. 12, p. 1211-1229.

Zazworka, N., Spínola, R., Vetro, A., Shull, F. e Seaman, C. (2013). **A Case Study on Effectively Identifying Technical Debt.** International Conference On Evaluation And Assessment In Software Engineering, p. 42-47.