Insper

Programa Institucional de Bolsas de Iniciação Científica (PIBIC)

Ciência da Computação

Modelos de Gestão de Dívida Técnica

Eduardo Takei Yaginuma

Orientador: Graziela Simone Tonin

São Paulo

2025

Programa Institucional de Bolsas de Iniciação Cientifica (PIBIC)

Eduardo Takei Yaginuma
Ciência da Computação

Modelos de Gestão de Dívida Técnica

Relatório final apresentado ao Programa Institucional de Bolsas de Iniciação Cientifica (PIBIC)

Orientador: Graziela Simone Tonin

São Paulo

2025

Resumo

Nos últimos anos, o impacto da dívida técnica tem se tornado cada vez mais evidente no desenvolvimento de software, afetando diretamente a qualidade dos produtos, a produtividade das equipes e os custos de manutenção. Empresas de diferentes setores estão reconhecendo que a dívida técnica não é apenas um problema técnico, mas também um fator estratégico que pode comprometer a inovação e a competitividade.

Diante desse cenário, este estudo tem como objetivo realizar uma revisão sistemática da literatura para identificar e analisar modelos de gerenciamento de dívida técnica em projetos de desenvolvimento de software. Partindo do modelo técnico de gestão de dívida proposto por Graziela Simone Tonin, a pesquisa busca explorar abordagens complementares adotadas por equipes ágeis, investigando diferentes metodologias, estratégias e ferramentas utilizadas no processo de gerenciamento da dívida técnica. O foco está em modelos que abrangem aspectos essenciais desse gerenciamento, incluindo identificação, classificação, monitoramento, priorização e pagamento da dívida técnica.

A partir dos estudos desenvolvidos na tese de Tonin G.S., a pesquisa pretende identificar outros modelos de gestão de dívida técnica, compará-los e prover novos *insights* a partir da análise de estudos recentes sobre essas abordagens. O estudo visa fornecer uma visão aprofundada sobre as melhores práticas no gerenciamento da dívida técnica no contexto do desenvolvimento ágil de software. A pesquisa também objetiva identificar tendências emergentes, como o uso de inteligência artificial e automação para otimizar a gestão da dívida técnica. Um dos propósitos é que essa revisão sistemática possa ser publicada em plataformas internacionais, contribuindo para a evolução das práticas de gerenciamento da dívida técnica na indústria de software.

Palavras-chave: Dívida Técnica; Gestão de Dívida Técnica; Modelo de Dívida Técnica; Qualidade de Software; Monitoramento de Dívida Técnica; Inteligência Artificial

Abstract

In recent years, the impact of technical debt has become increasingly evident in software development, directly affecting product quality, team productivity, and maintenance costs. Companies across different sectors are recognizing that technical debt is not merely a technical problem, but also a strategic factor that can compromise innovation and competitiveness.

Given this scenario, this study aims to conduct a systematic literature review to identify and analyze technical debt management models in software development projects. Building on the technical debt management model proposed by Graziela Simone Tonin, the research seeks to explore complementary approaches adopted by agile teams, investigating different methodologies, strategies, and tools used in the technical debt management process. The focus is on models that encompass essential aspects of this management, including identification, classification, monitoring, prioritization, and payment of technical debt.

Beyond reviewing Tonin G.S.'s thesis, the research intends to add new insights through the analysis of recent studies examining these approaches. The study aims to provide an in-depth perspective on best practices and innovations in technical debt management within the context of agile software development. The research also seeks to identify emerging trends, such as the use of artificial intelligence and automation to optimize technical debt management. One of the purposes is that this systematic review can be published on international platforms, contributing to the evolution of technical debt management practices in the software industry.

Keywords: Technical Debt; Technical Debt Management; Technical Debt Model; Software Quality; Technical Debt Monitoring; Artificial Intelligence.

Sumário

1. Dívida Técnica 6
   1.1 Definição 7
   1.2 Tipos de Dívida Técnica 8
   1.3 Consequências da Dívida Técnica
2. Modelos de Gestão de Dívida Técnica 9
   2.1 Identificação 9
   2.2 Monitoramento 9
   2.3 Estimativa 10
   2.2 Pagamento 10
3. Objetivos de Pesquisa 12
4. Metodologia 13
   4.1 Objetivos e Questões de Pesquisa 13
   4.2 String de Busca 14
   4.3 Seleção de Textos 14
   4.2 Utilização de Inteligências Artificiais 15
5. Modelos Existentes 17
6. Análise Comparativa dos Modelos 20
   6.1 Similaridades 20
      6.1.1 Ciclo de Gestão Abrangente 20
      6.1.2 Importância da Visibilidade e Conscientização 22
      6.1.3 Utilização de Ferramentas de Análise e Gestão 23
      6.1.4 Refatoração como Estratégia de Pagamento 24
      6.1.5 Envolvimento da Equipe 26
   6.2 Diferenças 28
      6.1.1 Foco Aprofundado em Causas e Motivações 28
      6.1.2 Análise Abrangente de Impactos 29
      6.1.1 Natureza Empírica e Iterativa com Adaptação Contínua 30
      6.1.2 Caracterização como Atividade Central 31
7. Oportunidades de Aprimoramento 32
8. Considerações Finais 34
   8.1 Trabalhos Futuros e Ameaças a Validade 35

Referências 37

1. Dívida Técnica

1.1 Definição

A dívida técnica é um conceito fundamental no campo de desenvolvimento de software que tem recebido atenção de grandes empresas e páginas como Oracle (2023), Medium (2023) e Microsoft (2024). Segundo Blumberg (2023), cerca de 30% dos CIOs entrevistados acreditam que mais de 20% do orçamento de TI destinado a novos produtos acaba sendo redirecionado para resolver problemas decorrentes da dívida técnica. Com a digitalização avançando rapidamente, cada vez mais empresas têm voltado sua atenção para essa questão.

A dívida técnica, pode ser definida como o resultado de escolhas inadequadas de *design* e implementação durante o desenvolvimento de software, que comprometem a compreensibilidade e a manutenção do sistema (Tufano *et al*., 2015). Essas decisões, frequentemente tomadas sob pressão de prazos apertados e pela urgência de entregar funcionalidades rapidamente, priorizam o tempo de lançamento no mercado em detrimento da qualidade do código. Como consequência, a dívida técnica se acumula ao longo do tempo, levando ao aumento da complexidade e custos associados à manutenção do software, além de dificultar sua evolução.

Além disso, o conceito de "dívida" é utilizado porque, quanto mais tempo a dívida técnica permanece sem ser resolvida, maior será o custo para corrigi-la. Por exemplo, se um código imaturo for introduzido no código e o desenvolvimento continuar sem que ele seja corrigido, novas partes do sistema poderão se tornar dependentes desse trecho defeituoso. Com o tempo, refatorar esse código exigirá alterações não apenas no código original, mas também em todas as seções que passaram a depender dele, aumentando significativamente o esforço e o custo da manutenção.

1.2 Tipos de dívida técnica

Cunningham, em seu relatório na conferência de OOPSLA (1992) apresentou sua primeira definição de metáfora de dívida técnica afirmando que escrever e lançar código imaturo rapidamente pode ser comparado a contrair uma dívida. Ele afirma que um pouco de "dívida técnica" pode acelerar o desenvolvimento, desde que seja paga posteriormente com uma reescrita do código. O problema surge quando essa dívida não é quitada. Cada momento gasto com um código mal estruturado equivale a pagar juros sobre essa dívida, tornando a manutenção cada vez mais difícil. Se essa situação se prolongar, toda a equipe de engenharia pode ficar travada sob o peso de uma implementação desorganizada, dificultando a evolução do sistema e comprometendo sua sustentabilidade a longo prazo. Para lidar com esse problema de forma eficaz, é importante compreender os diferentes tipos de dívida técnica e seus impactos no ciclo de vida do desenvolvimento.

O estudo de Li et. al (2014) realiza um mapeamento sistemático cujo um dos focos foi classificar a dívida técnica em 10 tipos. Segundo o estudo, os tipos de dívida técnica podem ser classificados com base nos aspectos do desenvolvimento que são comprometidos. A dívida de **requisitos**, por exemplo, surge quando há uma diferença entre o que foi especificado e o que foi realmente implementado, resultando em um software desalinhado com as necessidades do usuário ou do negócio. A dívida do tipo **arquitetural** decorre de decisões que prejudicam a escalabilidade e manutenção do sistema. A dívida de **design** ocorre quando a estrutura do software é comprometida por atalhos tomados na fase de concepção. Já a dívida de **código** se refere a códigos mal escritos, imaturos ou que não seguem boas práticas, dificultando a manutenção. A falta ou insuficiência de testes cria a dívida de **testes**, aumentando o risco de falhas. A dívida de ***build*** está relacionada a problemas na automação e nos *scripts* de compilação e implantação do software. Além disso, a dívida de **documentação** surge quando informações essenciais sobre o sistema estão desatualizadas ou ausentes. A dívida de **infraestrutura** ocorre quando a configuração de processos e ferramentas impacta negativamente a eficiência do time. Problemas no gerenciamento de versões caracterizam a dívida de **versionamento**, dificultando a evolução do código. Por fim, a dívida de **defeitos** engloba *bugs* e falhas que precisam ser corrigidos para garantir o funcionamento adequado do software.

1.3 Consequências da Dívida Técnica

A dívida técnica (TD) pode ter tanto efeitos positivos quanto negativos em um projeto de software, conforme discutido por Li *et al*. (2014). Quando inserida intencionalmente, a TD pode oferecer benefícios de curto prazo, como acelerar o desenvolvimento de novas funcionalidades e entregar o software mais rápido. No entanto, quando a TD é gerada de forma não intencional, sem que a equipe de desenvolvimento ou o gerente do projeto estejam cientes de sua existência ou das suas consequências, ela pode ser acumulada de forma invisível e não resolvida, o que resulta em desafios significativos nas tarefas de manutenção e evolução do software.

Por isso, à medida que se acumula, o processo de gerenciamento de dívidas técnicas se torna mais complexo. As tarefas de manutenção e evolução, nesse contexto, referem-se à necessidade de refatoração, correção de erros e adaptação do sistema a novos requisitos, que se tornam mais complicadas e caras quanto maior a quantidade de dívida técnica acumulada.

2. Modelos de Gestão de Dívida Técnica

Atualmente, há uma ampla variedade de artigos que exploram diferentes modelos de gestão da dívida técnica, testados em experimentos com empresas reais. Cada autor adota métricas próprias para definir o que constitui dívida técnica, seus tipos e formas de identificação. Além disso, os modelos propõem métodos distintos para gerenciá-la e apresentam os resultados obtidos nos experimentos. De forma geral, o processo de gestão da dívida técnica pode ser dividido em quatro etapas principais: identificação, monitoramento, estimativa e pagamento. Vale ressaltar que, embora essas etapas estejam presentes na maioria dos modelos, as métricas e métodos empregados podem variar significativamente

2.1 Identificação

A identificação da dívida técnica é um processo fundamental para seu gerenciamento eficaz. Brown *et al* (2010) menciona alguns pontos importantes a se considerar para identificação de dívidas técnicas. Em muitos casos, apenas um membro da equipe tem conhecimento de sua existência em determinada parte do código. Para avaliar corretamente a dívida técnica, é necessário compreender seu valor, que representa a diferença entre a versão atual do software e a versão ideal para entrega. O valor atual do software deve ser analisado considerando o impacto da dívida ao longo do tempo, já que seu acúmulo pode levar à desvalorização do código e dificultar sua manutenção. Além disso, é essencial identificar a origem da dívida, distinguindo se foi introduzida estrategicamente para acelerar entregas ou se decorreu de más práticas de desenvolvimento. Outro fator crítico é o impacto da dívida técnica, ou seja, quais áreas do sistema precisarão ser corrigidas para mitigar seus efeitos e evitar problemas futuros.

2.2 Monitoramento

O monitoramento da dívida técnica é essencial para evitar que ela seja esquecida e para acompanhar seu impacto ao longo do desenvolvimento do software. Tonin G.S. realizou um experimento no qual equipes de desenvolvimento foram introduzidas ao conceito de dívida técnica e tiveram que lidar com ela ao longo de seus projetos. Os resultados mostraram que algumas equipes adotaram um método simples e rápido, anotando apenas a identificação, o tipo e a localização da dívida, o que facilitou o acompanhamento sem comprometer o fluxo de trabalho. Além disso,

várias equipes relataram que a anotação foi essencial, pois, embora notasse a inserção de dívida técnica durante o desenvolvimento, frequentemente acabavam esquecendo dessas ocorrências posteriormente.

Esse estudo mostrou que um registro contínuo permite identificar se a dívida está se tornando um obstáculo para novas implementações, comprometendo a qualidade do código ou elevando os custos de manutenção. Além disso, é fundamental que toda a equipe esteja ciente dessas dívidas, pois elas podem influenciar diretamente a entrega de novas funcionalidades. Para garantir um controle eficiente, é possível utilizar desde métodos simples, como quadros com *post-its*, até soluções mais avançadas, como ferramentas de gestão de projetos.

2.3 Estimativa

Antes de pagar uma dívida técnica, é fundamental estimá-la corretamente para garantir que os esforços de remediação sejam eficazes e alinhados com os objetivos do negócio. Bill *et al*. (2012) sugere também alguns passos para estimar DT. O primeiro passo é definir prioridades estratégicas de qualidade para determinar quais aspectos do sistema devem ser priorizados. Em seguida, devem ser estabelecidas metas e planos para pagar as dívidas. Com base nessa análise, durante o planejamento de cada ciclo de lançamento ou *sprint*, a equipe de desenvolvimento pode priorizar quais dívidas devem ser corrigidas para melhorar a qualidade do código. A resolução da dívida ocorre como parte das atividades normais de desenvolvimento e manutenção, incluindo práticas como refatoração. Além disso, é essencial monitorar os resultados dessas ações, comparando-os com as metas estabelecidas e compartilhando o progresso com os desenvolvedores.

2.4 Pagamento

O pagamento da dívida técnica é a resolução dela, seja por refatoração, correção de defeitos ou outras ações. Mesmo após a estimativa, é crucial refletir sobre a decisão de pagar ou não essa dívida. Um trabalho de Holvitie et al. (2013) destaca que vários fatores influenciam essa decisão, como a gravidade da dívida, a existência de soluções alternativas, a urgência do pagamento, o esforço necessário para implementar a correção, o risco da solução proposta e o escopo de testes exigidos. Esses critérios permitem que as equipes avaliem se a dívida técnica deve ser resolvida imediatamente ou se pode ser adiada, levando em consideração tanto o impacto no desenvolvimento quanto às necessidades do cliente. Assim, o processo de pagamento da dívida técnica deve ser estrategicamente planejado, levando em

conta o custo e os benefícios da resolução, além do contexto do projeto.

3. Objetivos de Pesquisa

O objetivo desta pesquisa é analisar e comparar modelos existentes de gestão da dívida técnica, investigando diferentes métodos utilizados, critérios para priorização das dívidas, métricas de avaliação, entre outros aspectos. Além disso, ao revisar artigos mais recentes, será possível identificar novas tecnologias que possam facilitar esse processo, examinar como as empresas estão se organizando atualmente e compreender outros fatores que evoluem com o tempo. Ao comparar esses modelos com o proposto por Tonin.G.S, a pesquisa busca destacar semelhanças, diferenças e áreas potenciais para aprimoramento ou complementaridade. A intenção também é explorar as melhores práticas na gestão da dívida técnica e identificar lacunas que possam impulsionar avanços na área, como por exemplo a utilização de Inteligência Artificial.

4. Metodologia

A pesquisa envolveu um mapeamento sistemático sobre modelos de gestão da dívida técnica. O processo iniciou-se com a definição de um protocolo contendo um conjunto de regras para orientar a seleção de artigos, estabelecendo métricas relevantes como o objetivo da pesquisa, as perguntas de pesquisa, restrições de data de publicação e a exigência de artigos em inglês, definição de String de busca, critérios de inclusão e exclusão, entre outros critérios.

4.1 Objetivos e Questões de Pesquisa

Os objetivos da pesquisa foram:

- Identificar e analisar modelos de gestão da dívida técnica presentes na literatura;
- Avaliar as características e abordagens desses modelos na identificação, mensuração, priorização e resolução da dívida técnica;
- Comparar os achados com o modelo de Tonin, G.S., a fim de compreender semelhanças, diferenças e possíveis complementaridades;
- Examinar a literatura de engenharia de software em busca de *insights* sobre práticas de gestão da dívida técnica;
- Identificar lacunas na literatura e nas práticas atuais que possam direcionar pesquisas futuras ou modelos aprimorados;
- Explorar novas tecnologias desenvolvidas nos últimos anos que possam auxiliar na gestão da dívida técnica.

Para guiar a pesquisa, estabeleceu-se uma pergunta principal:

"Quais modelos existem para identificar, mensurar, priorizar e gerir dívida técnica em projetos de desenvolvimento de software e como times ágeis aplicam esses modelos?"

Além disso, foram definidas cinco questões de pesquisa específicas para orientar a extração de dados, a análise e a síntese das evidências:

- **RQ1:** Como os modelos de gestão da dívida técnica se integram às metodologias ágeis no desenvolvimento de software?
- **RQ2:** Quais estratégias e técnicas esses modelos empregam para identificar e monitorar a dívida técnica ao longo do tempo?
- **RQ3:** Como os modelos existentes medem e priorizam diferentes tipos de dívida

técnica e quais critérios utilizam para essa priorização?

- **RQ4:** Quais são as práticas comuns para o pagamento ou redução da dívida técnica nos modelos, e como elas são aplicadas dentro das equipes ágeis?

4.2 String de Busca

Um desafio importante foi a construção de uma *string* de busca eficiente, capaz de maximizar o retorno de artigos relevantes, ao mesmo tempo em que excluía materiais não pertinentes. A string inicial — ("Technical Debt") AND ("Identification" OR "Measurement" OR "Monitoring" OR "Payment" OR "Management") AND ("Agile Teams" OR "Scrum Teams") AND ("Model" OR "Framework") — trouxe resultados relevantes, porém em grande parte relacionados à gestão de times ágeis, e não à dívida técnica em si. Após diversos testes, a *string* foi refinada para:

('Technical Debt') AND ('Technical Debt Management' OR 'Technical Debt Administration' OR 'Technical Debt Identification') AND ('Agile Teams' OR 'Agile Development Teams' OR 'Agile Software Teams') AND ('Model' OR 'Framework') AND ('Identification' OR 'Prioritization' OR 'Management').

Os termos da *string* estão em inglês pois as principais *venues* que contém as principais publicações da área, são todas em inglês.

Com essa versão refinada, as buscas foram realizadas em quatro bases de dados científicas: ScienceDirect, ACM, Semantic Scholar e IEEE Xplore, além do uso de ferramentas de inteligência artificial, *Elicit e Scit,* utilizando a mesma *string*. A consistência e a sobreposição dos artigos retornados pelas diferentes fontes foram avaliadas, evidenciando a confiabilidade da *string* de busca

4.3 Seleção de Textos

A análise inicial indicou que a maioria dos trabalhos abordava três enfoques principais: propostas de modelos de gestão de dívida técnica, aplicação desses modelos em equipes de desenvolvimento ágil e estudos sobre como equipes ágeis lidam com a dívida técnica ao longo do ciclo de desenvolvimento. Também foram encontrados textos fora do escopo ou inconsistentes, como artigos sobre outros tipos de dívida ou com menções superficiais à dívida técnica.

Após a coleta, realizou-se um primeiro filtro considerando o ano de publicação e a leitura dos resumos (*abstracts*) para verificar a relevância dos artigos. Dos 49 papers obtidos inicialmente, desconsideramos 20, restando então 29 papers potencialmente relevantes para nossa pesquisa. Estes foram então compilados em uma tabela detalhada

contendo informações como: *Venue,* DOI, Direct URL, CAPES URL, título, abstract, palavras-chave, autor(es), ano de publicação, editora, periódico, tipo de estudo e país dos autores.

A seleção final considerou três critérios de inclusão:

- **I1:** Estudos primários, secundários ou terciários que descrevem modelos ou iniciativas (mesmo que não implementados ou avaliados) para identificar, monitorar e gerenciar a dívida técnica no desenvolvimento ágil de software;
- **I2:** Artigos escritos em inglês;
- **I3:** Artigos publicados nos veículos listados no protocolo.

E quatro critérios de exclusão:

- **E1:** Artigos que descrevem problemas relacionados à dívida técnica, mas não apresentam soluções ou modelos;
- **E2:** Modelos não desenvolvidos em equipes ágeis de software;
- **E3:** Artigos duplicados;
- **E4:** Publicações anteriores a 2018.

Essa verificação ocorreu em três etapas independentes:

1. Primeira verificação pelo aluno-pesquisador;
2. Segunda verificação pela professora orientadora;
3. Terceira verificação com apoio de uma ferramenta de inteligência artificial, garantindo validação cruzada e triangulação das informações, visando aumentar a confiabilidade em relação aos estudos selecionados e aos excluídos

Na etapa final, os 22 artigos selecionados foram analisados de forma aprofundada, com foco nos objetivos da pesquisa. Foram analisados os modelos de gestão da dívida técnica, suas características e abordagens na identificação, mensuração, priorização e resolução da dívida. Os resultados foram correlacionados com o modelo de **Tonin, G.S.**, permitindo compreender semelhanças, diferenças e possíveis complementaridades. Além disso, foram identificadas lacunas na literatura e avaliadas novas tecnologias que possam contribuir para uma gestão mais eficaz da dívida técnica.

4.4 Utilização de Inteligências Artificiais

Durante nossa pesquisa, nos deparamos com um grande volume de textos retornados, o que tornou o processo de análise detalhada desafiador. Para lidar com essa

quantidade significativa de informações, decidimos empregar o NotebookLM, uma inovadora ferramenta de inteligência artificial desenvolvida pelo Google. Essa tecnologia foi fundamental para nossa abordagem, pois possui a capacidade de analisar, sintetizar e extrair dados relevantes de documentos extensos de maneira eficiente. Além disso, enxergamos como uma oportunidade para testar o uso de IA para pesquisas científicas.

Antes de realmente utilizá-lo em nossa pesquisa, testamos a IA com artigos que já havíamos lido e fizemos perguntas a fim de verificar a confiabilidade das informações geradas e o resultado foi bastante positivo e coerente. O uso do NotebookLM nos permitiu filtrar os textos que realmente apresentavam propostas de modelos para a gestão de dívida técnica. Além disso, a ferramenta nos auxiliou a desvendar as diferentes etapas de gestão associadas a esses modelos, algo que teria sido difícil de realizar manualmente devido ao volume de dados. Ainda assim, mesmo com a revisão criteriosa de todos os textos analisados, reconhecemos que podem haver incoerências ou interpretações não totalmente alinhadas ao conteúdo original, uma vez que a ferramenta pode simplificar informações ou estabelecer relações que não foram explicitamente descritas pelos autores. Por isso, utilizamos a IA como um recurso de apoio, priorizando, sempre que possível, a checagem e a validação com as fontes primárias, através da dupla checagem de uma amostra aleatória dos estudo.[1]

[1] Tabela dos artigos retornados pela string: All papers returned by the search string.
Tabela dosartigos selecionados a partir da leitura do resumo e conclusão: Texts selected through the reading of the abstract and conclusion.
Lista final de artigos selecionados comparados usando NotebookLM: Paper Information.

5. Modelos Existentes

Foram considerados principalmente estudos primários, que envolvem a criação e/ou aplicação de modelos de gestão da dívida técnica e também estudos secundários e terciários, que revisam modelos existentes ou realizam sínteses de revisões anteriores. Além disso, também foram analisados materiais de portais e outras fontes para compreender o panorama atual da dívida técnica, incluindo novos conceitos, abordagens emergentes e tendências recentes.

O estudo “Technical Debt Management in the Context of Agile Methods in Software Development", conduzido por Graziela Simone Tonin (2018) representa um exemplo significativo de gestão estruturada da dívida técnica, combinando revisão teórica, experimentação prática e análise de dados em múltiplas etapas Esse modelo foi dividido em cinco etapas principais: revisão da literatura, estudo de caso em uma empresa de TI, experimentação no Extreme Programming Laboratory (XP Lab) da USP, um segundo experimento no XP Lab com um foco distinto e, por fim, um estágio no exterior para análise aprofundada dos resultados. Um aspecto relevante desse modelo é que ele combinou diferentes abordagens para testar a gestão de dívida técnica: desde a análise teórica até a implementação de métodos práticos dentro de empresas reais. Este estudo enfatiza a relevância de considerar o conceito de dívida técnica no processo de desenvolvimento, a fim de garantir a criação de software de alta qualidade. O modelo de gestão proposto visa não apenas identificar e monitorar, mas também gerenciar as dívidas técnicas existentes através de métodos e ferramentas adequadas. Além disso, o estudo oferece valiosos insights sobre os diferentes tipos de dívida técnica, estratégias de priorização e práticas eficazes para evitar a acumulação dessas dívidas ao longo do ciclo de desenvolvimento.

No início da pesquisa, alguns textos foram analisados para realizar uma revisão bibliográfica sobre modelos de gestão de dívida técnica existentes. Esses estudos forneceram *insights* complementares, consolidando conceitos fundamentais e diferentes abordagens metodológicas. Entre eles, Rios *et al*. (2018) oferecem uma visão ampla sobre tipos de dívidas, estratégias de gestão e tendências de pesquisa, enquanto Mendes (2024) apresenta a dívida técnica de forma didática, classificando-a em ingênua, inevitável e estratégica, e enfatizando suas consequências práticas para equipes e produtos. Esses trabalhos ajudaram a fundamentar a compreensão do tema e serviram como referência para a seleção e análise mais criteriosa de textos realizada posteriormente, seguindo o protocolo e os critérios definidos na metodologia desta pesquisa.

A etapa seguinte consistiu na identificação e seleção de estudos de forma rigorosa, com base no protocolo estabelecido, permitindo a análise de trabalhos mais recentes e relevantes. Essa abordagem aprofundou a compreensão dos modelos de gestão da dívida técnica e suas aplicações em contextos ágeis, fornecendo evidências adicionais para a discussão dos resultados.

Dentre os textos selecionados, destacam-se diferentes modelos e *frameworks* que oferecem perspectivas variadas para a gestão da dívida técnica. O artigo LTD (Less Technical Debt) *Framework* sugere um modelo de gestão de DT para equipes ágeis, cuja premissa central é aumentar a conscientização sobre a dívida técnica e integrá-la diretamente aos fluxos de trabalho do Scrum. O modelo se organiza em quatro atividades principais: *TD Consensus, TD Discovery, TD Planning e TD Payment*. A gestão é operacionalizada por meio da documentação das dívidas em forma de *TD Stories*, que são integradas ao backlog principal, garantindo maior visibilidade e priorização contínua. O estudo de caso realizado com duas equipes Scrum demonstrou que a adoção do LTD levou à criação de um artefato comum de registro, à alocação de tempo fixo para pagamento e à inclusão sistemática do tema nas cerimônias ágeis, resultando em redução efetiva da dívida técnica.

Já o estudo Analyzing the concept of technical debt in the context of agile software development: A systematic literature review (Behutiye *et al*. 2017), oferece uma síntese abrangente sobre causas, consequências e estratégias de gestão nesse contexto. Foram identificadas doze estratégias relevantes, sendo as mais significativas a refatoração contínua e o aumento da visibilidade por meio de ferramentas e indicadores. Outras práticas incluem a definição de uma *Definition of Done* (DoD)compartilhada, a priorização explícita das dívidas técnicas, o uso de análise de código automatizada e a aplicação de dashboards coletivos. O estudo também destaca lacunas importantes, como a carência de frameworks práticos validados e de estratégias baseadas em custo, além de reforçar a necessidade de integração entre ferramentas e metodologias ágeis para uma gestão mais eficaz.

A pesquisa “Lean and Agile Software Development for Managing” (Simangunsong *et al*., 2024), outra revisão sistemática, concentra-se na aplicação combinada de princípios Lean e metodologias ágeis para o gerenciamento da dívida técnica, especialmente em projetos de grande escala. O trabalho identifica três frameworks: o TAP Framework (leve e indicado para pequena escala), o próprio LTD Framework (adaptável a diferentes contextos) e o CoDVA Framework (*Continuous Debt Valuation Approach*), considerado mais robusto, mas também mais complexo por requerer o cálculo de um índice de

priorização baseado em fatores como tamanho da refatoração, benefícios esperados e dependências. A análise evidencia que a integração de práticas Lean, como a eliminação de desperdícios e a melhoria contínua, potencializa a eficiência da gestão de dívidas técnicas e a tomada de decisões baseadas em valor.

O artigo “Managing Technical Debt in Software Projects Using Scrum An Action” (Oliveira *et al*., 2015) descreve uma pesquisa-ação que adaptou o *framework* de Seaman e Guo (2011) para aplicação prática em equipes Scrum. O modelo revisado substituiu métricas probabilísticas complexas por medições baseadas no “principal” (esforço estimado para quitar a dívida) e no “valor atual dos juros” (esforço adicional gerado pela postergação). Essa abordagem permitiu maior objetividade e registro contínuo dos impactos da dívida. A participação de todos os papéis do Scrum na identificação e medição favoreceu uma visão mais ampla e colaborativa, enquanto a utilização de ferramentas de gestão de projetos como o Jira e o quadro de dívida técnica aumentou a visibilidade e ajudou na priorização de pagamento durante os sprints.

O estudo “Preventing technical debt with the TAP framework for Technical Debt Aware Management”, (Wiese *et al*., 2022) apresenta o TAP Framework, que se diferencia por integrar a gestão da dívida técnica diretamente ao gerenciamento de projetos, com foco tanto na prevenção quanto na quitação. As tarefas invisíveis ao cliente são classificadas em quatro grupos, incluindo os “*TD tickets*” para dívidas intencionais, cujo pagamento é planejado ainda no projeto de origem. O *framework* também define um contingente de tempo contínuo para manutenção e atualização tecnológica, buscando prevenir a acumulação de dívidas não intencionais. A atribuição clara de responsabilidades, a maior consciência organizacional e o pagamento oportuno são destacados como benefícios comprovados em sua aplicação prática.

Por fim, o artigo “Integrating Technical Debt Management and Software Quality Management Processes: A Normative Framework and Field Tests”, (2018) propõe um *framework* de processo normativo que integra a gestão da dívida técnica aos processos formais de gestão da qualidade de software. Estruturado em três etapas — tornar a dívida técnica visível, realizar análise de custo-benefício e controlar de forma contínua — o modelo adota artefatos específicos, como o registro de dívida técnica, que permitem rastrear a evolução e o impacto das dívidas. Essa abordagem considera a dívida técnica sob a ótica do custo da qualidade, permitindo análises econômicas mais precisas e controle estatístico do nível de dívida aceitável. Validado em três organizações distintas, demonstrou ser aplicável a equipes com diferentes portes e maturidade de processo.

6. Análise Comparativa dos Artigos

A gestão da dívida técnica (DT) é um campo em constante evolução na engenharia de software, com diversos modelos e abordagens propostos para mitigar seus impactos negativos. O modelo conceitual para gestão de dívida técnica desenvolvido por Tonin.G.S (2018) em sua tese, foi construído a partir de extensos estudos empíricos em ambientes acadêmicos e empresariais que oferecem uma perspectiva prática e iterativamente refinada. Esta seção se dedica a comparar o modelo de Tonin.G.S com outros modelos existentes, destacando suas semelhanças, diferenças e identificando áreas potenciais para aprimoramento ou complementaridade.

Diversos *frameworks* e abordagens para a gestão da dívida técnica foram propostos e avaliados na literatura. Ao comparar esses modelos com o proposto por Tonin.G.S, emergiram pontos de convergência e distinção importantes.

6.1 Similaridades (Pontos em Comum):

6.1.1 Ciclo de Gestão Abrangente

A gestão da dívida técnica (DT) é reconhecida como um elemento vital no desenvolvimento de software, e a maioria dos modelos para sua gestão converge para um ciclo de atividades similar. O objetivo desses modelos é prevenir crises desnecessárias e assegurar o sucesso a longo prazo, equilibrando o valor do software com os custos ao longo de seu ciclo de vida. Este alinhamento é observado, por exemplo, por Simangunsong *et al.* (2024).

O modelo conceitual proposto por Tonin (2018) é um exemplo abrangente dessa abordagem, estruturado em quatro áreas: impacto da DT, causas, contexto e etapas do processo de gestão. O modelo incorpora fases explícitas, como identificação, medição, monitoramento, priorização e pagamento, além da categorização da dívida.

As etapas descritas por Tonin (2018) são:

- Identificação: Consiste em compreender a metáfora da DT e coletar evidências, como *code smells* ou dados históricos, que podem ser registrados em ferramentas como o *Jira*.
- Categorização: categoriza as dívidas por tipo (arquitetura, defeitos, teste, outros), facilitando a gestão.
- Medição: Embora desafiadora, pode utilizar métricas específicas (risco e custo),

estimativas dos desenvolvedores e relatórios de ferramentas, conforme também observado por Oliveira *et al.* (2015).

- Monitoramento: Pode ser executado de forma manual (com quadros ou *pair programming*) ou automatizada (com ferramentas como SonarQube e Code Climate).

A etapa de priorização das dívidas técnicas busca determinar as mais críticas com base em critérios de impacto e custo, uma prática comum na maioria dos modelos para otimizar a alocação de recursos. Finalmente, o pagamento envolve refatoração ou a implementação de novas soluções, com a inclusão da dívida no backlog do *sprint*.

Esta estrutura de ciclo de gestão não é exclusiva do modelo de Tonin (2018). Outros *frameworks* também confirmam a validade dessa abordagem. O *framework* de Seaman e Guo (2011), que serviu de base para a pesquisa de Tonin, propõe as etapas de identificação, medição do esforço de pagamento e seleção para o ciclo de desenvolvimento. Da mesma forma, o LTD (*Less Technical Debt*) Framework de Xavier *et al*. (2017), um modelo que se propõe ser adaptável a metodologias ágeis, possui atividades-chave similares, como Consenso da DT, Descoberta da DT, Planejamento da DT e Pagamento da DT.

A semelhança entre os modelos é reforçada por outros trabalhos. O CoBeTDM de Harun *et al*. (2015) inclui identificação e avaliação, monitoramento, quantificação, priorização e pagamento. O *framework* de Yli-Huumo *et al.* (2016) também lista atividades como pagamento, prevenção, representação/documentação, identificação, medição, monitoramento, comunicação e priorização. Adicionalmente, o *framework* TAP (*Technical Debt Aware Project management*) de Wiese *et al*. (2021) integra a gestão da DT ao gerenciamento de projetos, focando na prevenção e quitação.

Outro elemento comum a todos esses modelos é a visibilidade da dívida técnica. A promoção da conscientização e a melhoria da comunicação entre as equipes e *stakeholders* são recorrentes, levando a decisões mais ponderadas e a uma percepção positiva na qualidade do *software*.

A análise dos modelos supracitados demonstra que, apesar das variações de abordagens e terminologias, há uma convergência significativa. O modelo de Tonin (2018) compartilha um conjunto essencial de etapas — identificação, medição, monitoramento, priorização e pagamento — com os *frameworks* de Seaman e Guo (2011), LTD (Xavier *et al*., 2017), CoBeTDM (Harun *et al.*, 2015), entre outros. Essa similaridade ressalta que uma abordagem estruturada para a gestão da dívida técnica não apenas aprimorando a

qualidade do software, mas também fortalecendo a comunicação e a tomada de decisões no contexto de desenvolvimento.

6.1.2 Importância da Visibilidade e Conscientização

A necessidade de tornar a dívida técnica (DT) visível e aumentar a conscientização sobre ela é um consenso fundamental na gestão de software, sendo um primeiro passo crucial para sua identificação, suas causas e subsequente gerenciamento explícito, conforme destacado por Kruchten *et al*. (2012). No contexto de métodos ágeis, onde a agilidade na entrega pode levar ao acúmulo de DT, a compreensão e gestão dessa dívida tornam-se vitais para a manutenção da qualidade, sustentabilidade e adaptabilidade a mudanças rápidas, como observado por Simangunsong *et al*. (2024).

O modelo conceitual de gestão da dívida técnica de Tonin (2018) enfatiza que a conscientização sobre a DT influencia diretamente o comportamento das equipes, promovendo maior reflexão e discussão sobre a qualidade do software. A aplicação deste modelo demonstrou que a visibilidade da DT melhora a comunicação da equipe, permite decisões mais conscientes e aprimora a percepção da qualidade geral do software. Equipes que se tornam conscientes da DT passam a pensar e discutir mais sobre a qualidade do software, focando na melhoria contínua, contribuindo para a construção de uma cultura de qualidade em toda a equipe.

Para operacionalizar essa visibilidade e conscientização, diversas ferramentas e abordagens visuais são amplamente sugeridas. O modelo de Tonin (2018), por exemplo, demonstrou a utilidade de quadros físicos de dívida técnica (*TD boards*) para documentar e tornar a DT visível. Além disso, o uso de ferramentas de gerenciamento de tarefas como Jira e a integração da DT em *Scrum boards* ou quadros *Kanban* facilitam sua inclusão nas atividades diárias da equipe. Monitores e com gráficos plotados, informando a saúde do código, também podem ser utilizados para monitorar e visualizar a quantidade de DT incorrida, incentivando as equipes a ações proativas.

O LTD (Less Technical Debt) Framework de XAVIER *et al.* (2017) adota outra abordagem para visibilidade e conscientização de DT, focando na criação de *TD Stories*, documentos semelhantes a *user stories*, mas dedicados a descrever instâncias de DT, e sugere que essas histórias sejam incorporadas ao *backlog* do produto para aumentar a conscientização e facilitar o planejamento do pagamento. O TAP (Technical Debt Aware Project Management) de WIESE *et al*. (2021) destaca o aumento da conscientização ao atribuir responsabilidades explícitas pela gestão da dívida técnica, o que leva a decisões mais conscientes e intencionais sobre a incoerência e quitação da

dívida.

Outras ferramentas que apoiam a visibilidade incluem SonarQube e Code Climate para análise de qualidade de código, e o uso de *dashboards* dedicados para visualizar a dívida. A implementação desses mecanismos não só facilita o monitoramento e o pagamento da DT, mas também promove um entendimento compartilhado dos desafios do software entre os *stakeholders*.

6.1.3 Utilização de Ferramentas de Análise e Gestão

A gestão da dívida técnica (DT) é amplamente apoiada pela utilização de ferramentas especializadas, buscando a integração com os processos de desenvolvimento de software para minimizar interferências e otimizar sua identificação e gerenciamento (Tonin, 2018; Simangunsongs *et al*, 2024; Wiese *et al.,* 2021). Essa abordagem reflete um consenso geral na área, onde a visibilidade da DT é considerada um primeiro passo crucial para seu gerenciamento explícito (Kruchten *et al*., 2012).

Para a análise e identificação de problemas no código, ferramentas de análise estática são amplamente empregadas. O *SonarQube* é frequentemente utilizado para avaliar a qualidade do código, identificar *code smells*, problemas de design, cobertura de testes, complexidade, *bugs* e vulnerabilidades, além de fornecer um índice específico de dívida técnica. Contudo, relatórios dessas ferramentas podem apresentar números tão elevados que não refletem o custo real da DT, levando as equipes a focar em métricas mais específicas (Tonin, 2018).

Outras ferramentas, como *Code Climate*, auxiliam na análise de repositórios e na mensuração de métricas como o *Grade Point Average* (GPA), que indicam qualidade e esforço de remediação (Tonin, 2018). PMD e Checkstyle também são citadas como soluções complementares para monitorar e identificar problemas de código (Tonin, 2018; Yli-Hummo *et al.,* 2016). O uso dessas ferramentas aumenta a conscientização sobre violações e ajuda a identificar *code smells* antes que se tornem dívidas incorporadas ao sistema (Al Mamun *et al*., 2019; Wiese *et al*., 2021).

Os sistemas de rastreamento de *issues* e tarefas são igualmente relevantes para documentar e gerenciar a DT. O *Jira* é amplamente usado para registrar e acompanhar itens de dívida técnica, que podem ser configurados como um tipo específico de *issue* ou *TD Story* e incorporados ao *backlog* (Tonin, 2018; Xavier *et al.*, 2017; OLIVEIRA *et al.,* 2015; YLI-HUUMO *et al.*, 2016). Esses registros podem conter informações como

criticidade, tempo estimado de implementação, localização no código e tarefas relacionadas, facilitando a priorização (Tonin, 2018). O *Trello* também é empregado como quadro *Kanban* para visualização e acompanhamento (Tonin, 2018; Oliveira *et al*., 2015; XAVIER *et al.*, 2017).

A prática de documentar DT em rastreadores de *issues* é comum, inclusive em projetos de código aberto no GitHub, onde *TD Stories* recebem rótulos específicos. A integração dessas histórias ao *backlog* unificado aumenta a visibilidade e facilita o gerenciamento paralelo das dívidas (Xavier *et al*., 2017).

Para reforçar a visibilidade, o modelo de Tonin (2018) demonstra a utilidade de quadros físicos (TD boards) para documentar e expor a DT. Além disso, monitores com gráficos são usados para monitorar a quantidade acumulada de dívida técnica e para mostrar métricas de qualidade de código, como cobertura de testes, incentivando ações proativas (Tonin, 2018). Dashboards dedicados também são recomendados, podendo impactar positivamente a conscientização dos desenvolvedores (Harun *et al*., 2015). A implementação desses mecanismos e a integração das ferramentas não apenas facilitam o monitoramento e o pagamento da DT, mas também promovem um entendimento compartilhado dos desafios do software entre equipes e stakeholders. Ao tornar a DT explícita e visível, as equipes passam a refletir mais sobre a qualidade, levando a decisões mais conscientes e a uma melhoria percebida na qualidade geral do código (Tonin, 2018; Wiese *et al.*, 2021). A visibilidade através da identificação e listagem dos itens de dívida técnica nos backlogs e quadros estimula o engajamento e o pagamento das dívidas, contribuindo para sua redução e para a construção de uma cultura de qualidade (Oliveira *et al.*, 2021; Tonin, 2018; Xavier *et al.,* 2017).

6.1.4 Refatoração como Estratégia de Pagamento

A refatoração é uma estratégia consistentemente mencionada e aplicada para o pagamento da dívida técnica (Behutiye *et al.*, 2024). A revisão sistemática de Behutiye *et al.* (2024), assim como o estudo de Tonin (2018) identifica a refatoração como uma das estratégias mais significativas para pagamento das dívidas e consequente gestão, sendo reconhecida como uma das atividades-chave em diversos modelos e frameworks de gestão da dívida técnica.

A refatoração consiste na reestruturação da base de código ou da arquitetura do sistema sem alterar o comportamento externo do software em desenvolvimento. Esta prática é bastante utilizada na gestão da DT em metodologias ágeis, com

recomendações que incluem educar os desenvolvedores, promover uma cultura de refatoração contínua e aplicar refatoração de forma constante sempre buscando a evolução e melhoria contínua do código e consequentemente suportando a evolução do produto de software.

No contexto dos modelos de gestão da DT, a refatoração é uma atividade central na fase de pagamento (ou *repayment*). O LTD Framework de Xavier *et al.* (2017) propõe a atividade de pagamento da DT, onde os desenvolvedores concentram-se em quitar as *TD Stories* selecionadas para o *sprint*, o que pode envolver a refatoração de código subótimo ou a correção de *workarounds* previamente identificados. O LTD sugere dedicar uma porcentagem do *sprint*, geralmente 20%, ao pagamento da dívida técnica, inspirado na “regra dos 20%” utilizada por empresas como o Google, permitindo que os desenvolvedores reservem parte do seu tempo para melhorias de sua escolha.

O modelo CoBeTDM de Harun *et al.* (2015) também inclui o *repayment* como uma de suas fases, onde itens selecionados na priorização são pagos por refatoração, seja a curto prazo (incorporados ao *backlog* atual) ou a longo prazo (mantidos para *backlogs* ou *releases* futuras).

A aplicação da refatoração é frequentemente impulsionada pela visibilidade da DT. Estudos de caso em ambientes acadêmicos e corporativos mostram que, ao tornar a dívida técnica explícita em quadros (*boards*), ferramentas de gestão de tarefas (como Jira e Trello) e relatórios de ferramentas de análise de código (como SonarQube e Code Climate), as equipes são incentivadas a refatorar o código e implementar outras melhorias de qualidade. A conscientização sobre a DT leva as equipes a refletir e discutir mais sobre a qualidade do software, favorecendo a alocação de tempo para refatoração e melhorias (Tonin, 2018; Wiese *et al.*, 2021).

Entretanto, a refatoração apresenta desafios e limitações:

- Pode não ser tão eficaz em projetos de grande escala quanto em projetos menores.
- É considerada demorada e difícil de aplicar em projetos complexos.
- Para dívidas de arquitetura, pode ser mais vantajoso aceitar o “juro” da dívida do que investir na refatoração imediata, caso o custo seja muito alto ou o sistema seja substituído em breve.
- A decisão de refatorar ou reescrever uma parte do código depende do tipo de sistema e do impacto de uma nova versão.

A experiência prática, conforme observada por Tonin (2018), demonstra que a refatoração é uma forma comum de pagamento da DT. Equipes incluíam a refatoração no *backlog* do *sprint* e a utilizavam como solução para os débitos identificados. Em projetos onde a dívida técnica era ignorada, a refatoração ou reescrita do código tornava-se inevitável para corrigir problemas e adaptar o negócio à tecnologia. A adoção de práticas como *pair programming* também se mostrou associada ao pagamento de itens de DT por meio da refatoração.

Em síntese, a refatoração é uma estratégia central e amplamente adotada para o pagamento da dívida técnica, especialmente quando esta é explicitamente identificada e gerenciada. Apesar dos desafios relacionados à sua aplicação, a integração da refatoração nos processos de desenvolvimento e o aumento da conscientização sobre a DT contribuem significativamente para a melhoria da qualidade do software a longo prazo.

6.1.5 Envolvimento da Equipe

O envolvimento de todos os papéis da equipe é um aspecto crucial e recorrente na gestão da dívida técnica (DT). Estudos demonstram que a identificação da DT é aprimorada quando todos os papéis do Scrum participam, enquanto a medição e a tomada de decisão são mais eficazes quando a equipe de desenvolvimento assume a responsabilidade (Tonin, 2018; Oliveira *et al.*, 2015).

Para a identificação da DT, pesquisas em empresas como a *SoftOne e SoftTwo* revelaram que todos os membros da equipe Scrum, incluindo Scrum Master, Product Owner e time de desenvolvimento podem contribuir significativamente. Por exemplo, um analista de negócios pode identificar uma falha em um processo que um desenvolvedor talvez não perceba. A participação de todos os papéis é relevante, pois a dívida técnica pode ser detectada sob diferentes perspectivas. Na SoftOne, 46 dívidas técnicas foram identificadas em um período de quatro semanas, e novos tipos de DT, como usabilidade, desempenho e infraestrutura, foram criados, evidenciando a diversidade da identificação (Tonin, 2018).

Embora ferramentas automáticas de análise de código ajudem a identificar certos tipos de dívidas, como *code smells*, a participação humana permanece essencial para detectar categorias de DT não capturadas por essas ferramentas (Tonin., 2018; Xavier *et al.,* 2017).

No que se refere à medição e priorização da DT, a equipe de desenvolvimento é considerada a mais adequada, pois possui a perspectiva técnica necessária e é responsável pelas atividades de codificação (Tonin, 2018). Em estudos práticos, houve adaptação das métricas para simplificar o processo: as três métricas probabilísticas originais (Principal, Quantidade de Juros Esperada e Probabilidade de Juros) foram reduzidas para duas — Principal (esforço estimado para pagar a DT) e Montante Atual de Juros (esforço extra necessário no futuro se a DT não for paga). Essa alteração aumentou a aceitação da equipe e do Product Owner (PO), que compreenderam a importância de monitorar e priorizar a DT durante o ciclo de desenvolvimento. Decisões de pagamento da DT eram tomadas em reuniões de *Sprint Planning*, considerando contexto e demanda, definindo se a dívida seria paga imediatamente ou mantida no *backlog* para monitoramento contínuo (Tonin, 2018).

A visibilidade da dívida técnica desempenha papel central no engajamento da equipe. Ao tornar a DT explícita e visível — por meio de quadros físicos ou virtuais, ferramentas de gerenciamento de tarefas como Jira e Trello, e relatórios de ferramentas de análise de código como SonarQube e Code Climate — as equipes são incentivadas a discutir e refletir sobre a qualidade do software. Essa conscientização melhora a comunicação e promove um entendimento compartilhado dos problemas. A presença de *TD Stories* nos *product* e *sprint backlogs* faz da dívida técnica uma preocupação de primeira ordem. O processo de discutir, ponderar a incorrência de DT e revisar a dívida ao longo do projeto contribui para a criação de uma cultura focada na melhoria contínua da qualidade do software (Tonin, 2018; Xavier *et al*., 2017; Wiese *et al*., 2021).

Entretanto, a implementação dessa colaboração apresenta desafios:

- A dificuldade inicial em estimar probabilisticamente as métricas da DT;
- O monitoramento manual da dívida pode ser demorado e requer dedicação constante, incentivando a busca por automação;
- A designação de apenas um membro para pagar a DT gerou desmotivação e acomodação dos demais, que poderiam inserir mais dívidas;
- Apesar de a integração da DT ao processo existente não exigir grande esforço, o tempo disponível para pagamento no *sprint* pode ser limitado, exigindo negociação com gerentes e líderes técnicos.

Apesar desses desafios, o envolvimento de toda a equipe e a visibilidade da DT mostraram-se fundamentais para promover uma cultura de qualidade e facilitar o gerenciamento eficaz da dívida técnica em projetos de desenvolvimento ágil (Tonin,

2018; Oliveira *et al*., 2015).

6.2 Diferenças (Aspectos Distintivos do Modelo de Tonin.G.S)

A seguir é relatada as principais diferenças identificadas dos modelos analisados em comparação com o modelo criado por Tonin (2018).

6.2.1 Foco Aprofundado em Causas e Motivações

Um diferencial do modelo conceitual de gestão da dívida técnica (DT) proposto por Tonin (2018) é o foco aprofundado na identificação e análise das causas e motivações que levam à sua contração. Essa abordagem vai além da simples constatação da dívida no código, buscando compreender os fatores subjacentes que a originam. Enquanto outros modelos, como o de Avgeriou *et al.* (2015), tendem a listar causas de forma descritiva, o modelo de Tonin (2018) trata este mapeamento como ponto de entrada central para todo o processo de gestão da DT

.A pesquisa de Tonin (2018) explora uma ampla gama de motivações, tanto visíveis quanto “invisíveis”, que podem resultar na ocorrência da DT. Entre elas estão pressões por entregas rápidas (time-to-market), imposições da diretoria sem participação da equipe, lacunas de conhecimento técnico ou inexperiência de desenvolvedores juniores, bem como a influência de especialistas que direcionam para soluções subótimas sob justificativas questionáveis. Também são incluídos fatores como necessidades de desempenho do sistema, restrições organizacionais, regras de segurança, inconsistências de dados, além de aspectos humanos e culturais, como a falta de documentação adequada. Ao trazer à tona essas múltiplas motivações, o modelo não apenas amplia a compreensão da origem das dívidas, mas também evidencia como elas se entrelaçam com dimensões técnicas, organizacionais e humanas.

Essa perspectiva se diferencia de outros modelos que concentram sua atenção principalmente nas propriedades da dívida ou nos artefatos do código. Enquanto o modelo de Avgeriou *et al.* (2015) reconhece causas, mas não as insere como parte estrutural da gestão, o modelo de Tonin (2018) as posiciona como componente fundamental dentro de um ciclo que integra identificação, medição, monitoramento, caracterização e pagamento. Essa abordagem ressalta o papel das causas e motivações não apenas como elementos de diagnóstico, mas como eixo estruturante para todo o processo de gestão da dívida técnica.

6.2.2 Análise Abrangente de Impactos

O modelo de Tonin (2018) diferencia-se por propor uma análise abrangente dos impactos da dívida técnica (DT), contemplando não apenas os efeitos no nível do código, mas também suas repercussões no projeto, na equipe e no negócio. Essa visão holística contrasta com abordagens que tendem a restringir os impactos a dimensões mais técnicas, como a qualidade do código ou a complexidade arquitetural. Para Tonin, compreender os efeitos em múltiplas esferas é essencial, pois a DT não gerenciada pode comprometer não apenas a manutenção do software, mas também a sustentabilidade do projeto e a competitividade organizacional.

No nível do projeto, a pesquisa identificou que a DT pode gerar atrasos expressivos, chegando a ultrapassar 120 dias em determinados casos, o que resulta em custos adicionais e aumento do chamado “juro” associado ao retrabalho. Já em relação ao negócio, destacam-se perdas de mercado e insatisfação de clientes, que podem comprometer diretamente a posição da empresa em seu setor. No âmbito da equipe, os impactos incluem desgaste, estresse e redução da motivação, com queda na produtividade e aumento da pressão por resultados.

Outros modelos, embora também abordem os impactos da DT, o fazem de maneiras distintas. O estudo de Yli-Huumo *et al.* (2016), por exemplo, dá maior ênfase a consequências de longo prazo, como a necessidade de horas extras constantes, a elevação da taxa de erros e o aumento da complexidade do sistema ao longo do tempo, o que torna a dívida cada vez mais difícil de ser controlada. Já Oliveira *et al.* (2015) destacam efeitos no nível da equipe, ressaltando que o gerenciamento explícito da DT contribui para melhorar a comunicação, fortalecer a reflexão sobre qualidade e favorecer a implementação de ações preventivas no processo de desenvolvimento.

Enquanto muitos desses modelos tendem a recortar os impactos em dimensões específicas — seja técnica, organizacional ou humana —, o modelo de Tonin (2018) os integra em uma perspectiva unificada, oferecendo um panorama completo para apoiar decisões estratégicas. Essa abordagem amplia a visibilidade da dívida técnica, permitindo que ela seja tratada não apenas como um problema técnico a ser resolvido, mas como um fator de gestão que influencia a qualidade contínua do software e a relação transparente com stakeholders.

6.2.3 Natureza Empírica e Iterativa com Adaptação Contínua

O modelo conceitual de gestão da dívida técnica (DT) proposto por Tonin (2018) tem como característica central sua natureza empírica e iterativa, desenvolvida a partir de múltiplos estudos de caso e ciclos de pesquisa-ação. Diferentemente de modelos essencialmente teóricos ou derivados de revisões sistemáticas, o modelo de Tonin (2018) nasce diretamente da prática, fundamentado em dados concretos obtidos em contextos reais de desenvolvimento de software. A pesquisa envolveu cenários variados, incluindo duas edições do curso XP Lab da Universidade de São Paulo, com nove equipes acadêmicas, e duas grandes empresas brasileiras de software, contemplando cinco equipes de negócio. Essa diversidade proporcionou uma base sólida para observar a ocorrência, o monitoramento e os impactos da DT em ambientes ágeis reais.

A abordagem iterativa do modelo ficou evidente nos três ciclos de pesquisa-ação realizados na Empresa B, nos quais a proposta inicial foi continuamente ajustada conforme as necessidades e dificuldades observadas nas equipes. Já na Empresa A, o estudo foi conduzido em duas fases complementares: a primeira voltada à investigação de causas e impactos da DT e a segunda dedicada à criação de mecanismos de identificação e monitoramento que pudessem se integrar naturalmente aos processos já existentes. Um princípio orientador foi evitar rupturas nos fluxos de trabalho das equipes, utilizando ferramentas e práticas já adotadas, como o Jira e quadros Scrum, de modo que o gerenciamento da DT fosse incorporado à rotina.

Essa natureza prática e adaptável diferencia o modelo de Tonin (2018) de outros já consolidados na literatura. O modelo de Avgeriou *et al*. (2015), por exemplo, é estruturado a partir de uma taxonomia teórica da dívida técnica, oferecendo uma visão abrangente, mas menos validada empiricamente em diferentes contextos industriais. Da mesma forma, *frameworks* como o de Yli-Huumo *et al.* (2016) propõem atividades detalhadas de gestão, mas apresentam menor ênfase na validação contínua em cenários reais e na adaptação às práticas cotidianas das equipes. Em contraste, o modelo de Tonin (2018) foi desenvolvido a partir da experiência direta de equipes, evoluindo de forma colaborativa e empírica, característica que o torna flexível, aplicável e continuamente ajustável.

O resultado desse processo é um modelo pragmático, que não apenas auxilia no gerenciamento da dívida existente, mas também fornece uma base para compreender suas causas, adaptar estratégias conforme as necessidades de cada projeto e incorporar a gestão da DT como parte integrante da melhoria contínua da qualidade do software.

6.2.4 "Caracterização" como Atividade Central

No modelo de gestão da dívida técnica (DT) de Tonin (2018), a etapa de Caracterização assume papel central e integrador, reunindo as atividades de medição e monitoramento da dívida em um único processo contínuo. Essa proposta contrasta com outros modelos da literatura que tendem a segmentar essas atividades. O modelo de Harun *et al.* (2015), por exemplo, distribui o monitoramento, a quantificação e a priorização em etapas distintas, enquanto o *framework* de Yli-Huumo *et al.* (2016) organiza medição e monitoramento como tarefas separadas dentro de suas oito atividades de gestão. Em contraponto, a unificação proposta por Tonin (2018) otimiza o processo ao tratar essas dimensões como partes interdependentes, fornecendo informações consistentes para decisões de pagamento, mitigação ou prevenção da DT.

A abordagem de Tonin (2018) também se diferencia no tratamento das métricas. O modelo inicial de Seaman e Guo (2011), que inspirou a pesquisa, recomendava indicadores como Principal (custo de eliminação da dívida), Probabilidade de Juros e Valor do Juro. Contudo, a aplicação prática demonstrou que métricas probabilísticas eram difíceis de estimar sem dados históricos robustos, gerando pouca aderência por parte das equipes. E Tonin (2018) propôs uma simplificação pragmática: concentrar-se em dois elementos centrais — Principal (esforço estimado para quitar a dívida) e Juro Acumulado (custo real incorrido sempre que a dívida afeta o desenvolvimento). Essa adaptação aumentou a aceitação do modelo e a clareza na priorização, alinhando os cálculos de esforço e custo às metas técnicas e de negócio de cada projeto.

O monitoramento, integrado à Caracterização, combina técnicas manuais e automatizadas. O modelo proposto e aplicado por Tonin aplica, em sua maioria, técnicas manuais enquanto outros modelos optam por técnicas digitais. As manuais incluem quadros físicos ou digitais (Scrum boards ou quadros específicos de DT), listas de dívidas técnicas, práticas de pair programming e discussões em Sprint Planning Meetings. Já as automatizadas envolvem o uso de ferramentas como Jira (para registro de itens de DT), SonarQube e Code Climate (análise estática de código, cobertura de testes, complexidade e duplicação), além de *plugins* específicos para monitoramento da qualidade em *merges*. A visualização é reforçada com gráficos em tempo real exibidos em *dashboards* e telas de TV, favorecendo a transparência.

7. Oportunidades de Aprimoramento

O modelo conceitual para gestão de dívida técnica proposto por Tonin (2018), resultado de estudos empíricos em ambientes acadêmicos e corporativos, oferece uma abordagem prática e baseada em dados para auxiliar equipes ágeis a documentar, monitorar e gerenciar a dívida técnica. Ele se estrutura em duas partes principais: identificação da dívida técnica e gestão das dívidas identificadas, incorporando as causas, impactos e o contexto de sua ocorrência. Embora seja um avanço significativo, a comparação com outros modelos na literatura revela pontos onde o modelo de Tonin poderia ser aprimorado ou complementado para oferecer uma solução ainda mais robusta e abrangente.

Uma área de aprimoramento surge da formalização da identificação e medição da dívida técnica. Enquanto o modelo de Tonin (2018) se baseia na compreensão da metáfora da dívida técnica pela equipe e na coleta de evidências, frequentemente por meio de discussões e relatórios de ferramentas, o framework CoBeTDM de Harun *et al.* (2015) sugere uma forma específica para coleta de dados que relaciona a dívida a *code smells* e *architectural smells* desde a fase de identificação e avaliação. Integrar um formulário de coleta de dados mais padronizado que sistematicamente capture e vincule a dívida a esses tipos de "cheiros" específicos poderia tornar a fase de identificação no modelo de Tonin mais rigorosa e menos dependente de uma análise humana inicial. Além disso, a quantificação da dívida em Harun *et al.* (2015) é fortemente baseada em uma análise de custo-benefício (ROI), o que poderia complementar a abordagem de Tonin (2018) de medir a dívida com base em métricas e impactos definidos pela equipe, oferecendo um critério mais objetivo para a priorização, especialmente em projetos de grande escala.

Outro ponto de melhoria está na incorporação explícita de "sintomas" e propriedades detalhadas da dívida técnica. O modelo conceitual de Avgeriou *et al*. (2015) foca nos elementos e nas relações da dívida técnica, identificando "sintomas" que ajudam a reconhecer a dívida, como a perda de qualidade, e suas propriedades, como principal e juros. Embora Tonin (2018) mencione evidências e propriedades da dívida, como visibilidade, principal e juros, uma integração mais formal e detalhada dos "sintomas" de Avgeriou *et al.* (2015) no processo de Tonin poderia enriquecer a compreensão da dívida pelas equipes e guiar melhor a coleta de informações sobre sua manifestação no sistema. Isso permitiria que as equipes não apenas identifiquem a dívida pela sua causa ou impacto, mas também pelos seus sinais visíveis no dia a dia do desenvolvimento.

O aspecto da prevenção também representa uma oportunidade de aprimoramento.

O framework de Huumo *et al.* (2016) destaca explicitamente a prevenção como uma atividade de gestão de dívida técnica, classificando-a em diferentes níveis de organização. Embora Tonin (2018) mencione a prevenção da dívida técnica e a importância de identificar motivações para mitigar recorrências, seu modelo não a apresenta como uma etapa ou atividade cíclica distinta na mesma proeminência que outras fases. Adicionar um componente mais formal de prevenção, talvez com diretrizes ou práticas específicas para evitar a acumulação de dívida, como o treinamento em tecnologias específicas para mitigar deficiências de conhecimento que Tonin (2018) identifica como causas, poderia fortalecer a proatividade do modelo.

Em termos de abrangência e especialização, o modelo de Lenarduzzi *et al.* (2021) se concentra especificamente na Dívida Técnica Arquitetural (ATD), incluindo listas de verificação (*checklists*) e impedimentos relacionados a esse tipo de dívida. Enquanto o modelo de Tonin (2018) é mais genérico, aplicando-se a todos os tipos de dívida técnica, ele poderia ser aprimorado ao integrar a capacidade de fornecer orientações mais granulares e específicas para diferentes categorias de dívida técnica (como arquitetural, design, infraestrutura, outros), baseando-se em checklists ou critérios adaptados para cada tipo. Isso ofereceria às equipes um guia mais detalhado, permitindo que o modelo seja flexível para diferentes contextos de projeto e tipos de dívida.

Por fim, a comunicação com o negócio pode ser mais formalizada. Embora o modelo de Tonin (2018) enfatize a importância de tornar a dívida técnica visível para o *Product Owner* e outras partes interessadas no negócio, e suas pesquisas demonstrem o impacto disso, o ciclo de gestão de dívida técnica de Bill *et al*. (2012) sugere explicitamente uma etapa de "Reportar ao Negócio". Esta etapa final poderia ser uma adição valiosa ao modelo de Tonin (2018), garantindo que os resultados e esforços de gestão da dívida técnica sejam continuamente comunicados e justificados em um nível estratégico, o que pode facilitar a alocação de recursos e a tomada de decisões de alto nível para o pagamento da dívida.

Em síntese, o modelo de Tonin (2018) estabelece uma base sólida e testada empiricamente para a gestão da dívida técnica em equipes ágeis. No entanto, ao considerar as abordagens complementares de outros frameworks, ele poderia ser enriquecido com uma identificação e medição mais formalizada e baseada em dados específicos (como *code/architectural smells* e ROI), uma integração mais profunda dos sintomas da dívida técnica, a inclusão explícita de estratégias de prevenção, diretrizes mais específicas para diferentes tipos de dívida, e uma formalização mais clara da comunicação estratégica com o negócio, tornando-o ainda mais abrangente e adaptável a diversos cenários de desenvolvimento de software.

8. Considerações Finais

A dívida técnica consolidou-se como um fator estratégico crucial no desenvolvimento de software, impactando diretamente a qualidade dos produtos, a produtividade das equipes e os custos de manutenção. Empresas de diversos setores reconhecem que sua influência vai além da esfera técnica, afetando também a capacidade de inovação e a competitividade no mercado. Nesse contexto, o presente estudo teve como objetivo realizar um mapeamento sistemático da literatura para identificar e analisar modelos de gerenciamento de dívida técnica em projetos de software, partindo do modelo proposto por Tonin (2018) e comparando-ocom abordagens complementares presentes na literatura. A pesquisa buscou compreender diferentes metodologias, estratégias e ferramentas empregadas no processo de gestão da dívida técnica, com foco em aspectos essenciais como identificação, classificação, monitoramento, priorização e pagamento.

A análise dos modelos investigados, muitos deles validados em empresas reais, evidencia um ciclo de gestão recorrente que envolve as etapas de identificação, monitoramento, estimação e pagamento. Entre as práticas mais eficazes destacam-se: a visibilidade contínua da dívida técnica para todos os envolvidos, utilizando ferramentas visuais e sua integração nos backlogs; a integração com metodologias ágeis já existentes, como Scrum, para assegurar a sustentabilidade do processo; e o envolvimento multidisciplinar da equipe, engajando desenvolvedores, arquitetos e Product Owners na identificação e tratamento da dívida. A medição pragmática e contextualizada também se mostra fundamental, recorrendo a métricas acionáveis como esforço para pagar o "principal" e o impacto associado aos "juros". A priorização guiada por custo-benefício e impacto no negócio, o pagamento contínuo ao longo das sprints e a documentação consistente das dívidas completam esse conjunto de boas práticas que sustentam a efetividade da gestão.

Dentro desse panorama, o modelo de Tonin (2018) se destaca por sua natureza empírica, iterativa e adaptativa, fruto de extensos estudos em contextos acadêmicos e corporativos. Seu diferencial reside na ênfase em minimizar interferências nos processos já utilizados pelas equipes, propondo uma integração flexível com ferramentas e práticas existentes. Outro aspecto notável é o foco nas causas e motivações da dívida técnica, considerando pressões de negócio, mudanças tecnológicas e decisões de curto prazo que comprometem a qualidade. O modelo também amplia a compreensão do problema ao analisar os impactos da dívida técnica em múltiplas dimensões – código, projeto, equipe, negócio e relação com clientes – fornecendo evidências que justificam investimentos para sua mitigação. A atividade de “Caracterização”, que combina medição, monitoramento e

ranqueamento com base em métricas priorizadas para cada projeto, representa um ponto central de inovação, tornando o modelo de Tonin altamente aplicável.

Apesar de sua abrangência e solidez, a comparação com outros *frameworks* evidencia lacunas e oportunidades de evolução. A formalização de métodos mais estruturados para identificação e medição de dívida técncia como a integração de *code smells*e *architectural smells* poderia tornar a coleta de dados menos dependente de julgamento humano (Avgeriou *et al*., 2015),. A inclusão explícita de um ciclo de prevenção sistemática (Huumo *et al.*, 2016; Paper Information) fortaleceria o caráter proativo do modelo. Diretrizes mais específicas para diferentes tipos de dívida (Lenarduzzi *et al.,* 2018) poderiam aumentar sua granularidade, enquanto a formalização de uma etapa de comunicação estratégica com o negócio (Bill *et al.*, 2012) potencializaria a visibilidade e o alinhamento organizacional.

Em síntese, o modelo de Tonin (2018) constitui uma base completa, prática e empiricamente validada para a gestão da dívida técnica em ambientes ágeis. Ainda assim, ao ser complementado por abordagens presentes na literatura, pode evoluir para uma proposta mais robusta, detalhada e adaptável, capaz de oferecer tanto suporte operacional às equipes quanto orientação estratégica para organizações que buscam equilibrar velocidade de entrega, qualidade de software e sustentabilidade no longo prazo.

8.1 Trabalhos Futuros e Ameaças a Validade

Durante o desenvolvimento dessa pesquisa, exploramos a utilização de IAs específicas para extração e análise de artigos. Isso acelerou o processo de pesquisa e nos permitiu analisar muito mais textos do que conseguiríamos fazendo esse processo de forma manual. No entanto, sabemos que concomitantemente essas ferramentas podem acabar perdendo informações valiosas e até mesmo levantar informações incoerentes. Por isso, primeiramente, um trabalho futuro que desejamos fazer é fazer uma análise mais rigorosa e explorar mais essas ferramentas para garantir a veracidade dessas informações e facilitar o processo de pesquisa.

Além disso, para realizar uma análise mais crítica e detalhada, pretendemos destacar diferenças e similaridades entre os modelos de forma mais visual, utilizando gráficos e tabelas. Isso tudo nos auxiliará a garantir uma análise mais completa e rigorosa dos modelos de gestão de dívida técnica existentes e extrair mais insights e práticas que podem agregar para a pesquisa.

Por fim, planejamos desenvolver um modelo unificado que sintetize todos os pontos

levantados ao longo desta pesquisa. A criação desse modelo proporcionaria uma estrutura integrada que poderia ser aplicada em diversos contextos de gestão de dívida técnica. Nosso objetivo é testar essa abordagem em empresas reais, especialmente aquelas que operam com equipes de desenvolvimento ágil. Este teste permitiria não apenas avaliar a eficácia e aplicabilidade do modelo em ambientes práticos, mas também colher feedback direto dos profissionais envolvidos, possibilitando ajustes e refinamentos

Referências

AL MAMUN, Md Abdullah; MARTINI, Antonio; STARON, Miroslaw; BERGER, Christian. Evolution of Technical Debt: An Exploratory Study. In: INTERNATIONAL WORKSHOP ON SOFTWARE MEASUREMENT AND INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS AND PRODUCT MEASUREMENT (IWSM MENSURA), 2019, Haarlem, Netherlands. 2019. Disponível em: https://www.researchgate.net/publication/336367478_Evolution_of_Technical_Debt_An_Exploratory_Study. Acesso em: 25 ago. 2025.

BEHUTIYE, T.; ISLAM, S.; LAGOS, P.; et al. Technical debt management: A systematic literature review. Journal of Systems and Software, v. 92, p. 137–158, 2024. Disponível em: [URL]. Acesso em: 14 ago. 2025.

BINSTOCK, Andrew. Reduce technical debt by valuing comments as much as code. Java Magazine, 18 jan. 2023. Disponível em: https://blogs.oracle.com/javamagazine/post/java-comments-reduce-technical-debt. Acesso em: 23 fev. 2025.

BLUMBERG, Sven. Demystifying digital dark matter: A new standard to tame technical debt. McKinsey, 23 jun. 2022. Disponível em: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/demystifying-digital-dark-matter-a-new-standard-to-tame-technical-debt. Acesso em: 28 fev. 2025.

BROWN, N. et al. Managing technical debt in software-reliant systems. In: FSE/SDP WORKSHOP ON FUTURE OF SOFTWARE ENGINEERING RESEARCH, 2010, ACM, 2010. p. 47–52.

CUNNINGHAM, W. Experience report-the why cash portfolio management system. 01 dez. 1992. 2 p. Disponível em: https://dl.acm.org/doi/10.1145/157710.157715. Acesso em: 25 ago. 2025.

CURTIS, Bill; SAPPIDI, Jay. Estimating the Principal of an Application's Technical Debt. IEEE Software, v. 29, n. 6, p. 34-42, nov. 2012. Disponível em: https://doi.org/10.1109/MS.2012.156. Acesso em: 26 ago. 2025.

HARUN, A.; Lichter et al. Towards a Technical Debt Management Framework based on Cost-Benefit Analysis, nov. 2015. Disponível em: https://www.researchgate.net/publication/284156853_Towards_a_Technical_Debt_Management_Framework_based_on_Cost-Benefit_Analysis. Acesso em: 14 ago. 2025.

HOLVITIE, J.; LEPPÄNEN, V. DebtFlag: Technical debt management with a development environment integrated tool. In: INTERNATIONAL WORKSHOP ON MANAGING TECHNICAL DEBT (MTD), 4., 2013, San Francisco, CA, USA. IEEE, 2013. p. 20-27.

KRUTCHEN, P.; LAGO, P.; VAN VLIET, H.; et al. Social debt in software engineering: Insights from industry. Empirical Software Engineering, dez. 2015. Disponível em: https://www.researchgate.net/publication/277355779_Social_debt_in_software_engineering_insights_from_industry. Acesso em: 14 ago. 2025.

LENARDUZZI, Valentina et al. A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools. Journal of Systems and Software, v. 171, 110827, janeiro 2021. Disponível em: https://doi.org/10.1016/j.jss.2020.110827. Acesso em: 25 ago. 2025.

LI, Zengyang et al. A systematic mapping study on technical debt and its management. The Journal of Systems and Software, v. 103, p. 423-439, maio 2015. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121214002854. Acesso em: 03 mar. 2025.

LI, Zengyang; AVGERIOU, Paris; LIANG, Peng. A systematic mapping study on technical debt and its management. Journal of Systems and Software, v. 101, p. 193-220, março 2015. Disponível em: https://doi.org/10.1016/j.jss.2014.12.027. Acesso em: 25 ago. 2025.

MC CONNELL, S. Code Complete: A Practical Handbook of Software Construction. 2. ed. Redmond: Microsoft Press, 2004. 960 p. Disponível em: https://www.semanticscholar.org/paper/Code-complete-a-practical-handbook-of-software-2nd-McConnell/7f6bdb113cd1364e12586216b89cae2d01f4d36e. Acesso em 23 ago. 2025.

MENDES, Pedro. Dívida Técnica: o que é e como resolvê-la. HNZ, 01 fev. 2024. Disponível em: https://hnz.com.br/divida-tecnica-o-que-e-e-como-resolve-la-copy/. Acesso em: 27 fev. 2025.

MICROSOFT. Understanding technical debt. Microsoft, 01 fev. 2024. Disponível em: https://www.microsoft.com/en-us/microsoft-365-life-hacks/organization/understanding-technical-debt. Acesso em: 25 fev. 2025.

OLIVEIRA, Frederico; GOLDMAN, Alfredo; SANTOS, Viviane Martins Rebello dos. Managing Technical Debt in Software Projects Using Scrum: An Action Research. In: AGILE CONFERENCE, 2015. IEEE, 2015. DOI:10.1109/Agile.2015.7. Acesso em: 16 ago. 2025.

RAMASUBBU, Narayan; KEMERER, Chris. Integrating Technical Debt Management and Software Quality Management Processes: A Framework and Field Tests. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE '18), 40., 2018. p. 883. Disponível em: https://doi.org/10.1145/3180155.3182529. Acesso em: 25 ago. 2025.

RECULPITO, Gilberto et al. Technical debt in AI-enabled systems: On the prevalence, severity, impact, and management strategies for code and architecture. Journal of Systems and Software, v.216, out. 2024. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121224001961. Acesso em: 09 mar. 2025.

RIOS, Nicole et al. 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, 2018. Disponível em: https://www.sciencedirect.com/science/article/pii/S0950584918300946. Acesso em: 01 mar. 2025.

SEAMAN, C.; GUO, Y. Chapter 2 - Measuring and Monitoring Technical Debt. In: Advances in Computers. v. 82, 2011. p. 25-46. Disponível em: https://doi.org/10.1016/B978-0-12-385512-1.00002-5. Acesso em: 14 ago. 2025.

SIMANGUNSONG, Surya Seven Y.; RAHARJO, Teguh; FITRIANI, Nur Afika. Lean and Agile Software Development for Managing Technical Debt on A Large-scale Software: A Systematic Literature Review. Indonesian Journal of Computer Science, v. 12, n. 6, 2024. Disponível em: https://doi.org/10.33022/ijcs.v12i6.3612. Acesso em: 16 ago. 2025.

TAURION, Cezar. Dívida Técnica (Technical Debt): como isso impacta seu negócio?. Medium, 08 jan. 2023. Disponível em: https://c-taurion.medium.com/dívida-técnica-technical-debt-como-isso-impacta-seu-negócio-3a72b86e733a. Acesso em: 22 fev. 2025.

TONIN, G. S. Technical Debt Management in the Context of Agile Methods in Software Development. 2018. 190 páginas. Dissertação (Mestrado em Engenharia de Software) – Universidade de São Paulo, São Paulo, 2018. Disponível em: https://www.teses.usp.br/teses/disponiveis/45/45134/tde-30072018-142720/publico/GrazielaSimoneToninThesis.pdf. Acesso em: 02 mar. 2025.

TUFANO, M. et al. When and Why Your Code Starts to Smell Bad. In: IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 37., 2015, Florence, Italy. IEEE/ACM, 2015. p. 403-414.

WIESE, Marion et al. Preventing technical debt with the TAP framework for Technical Debt Aware Management. [Título da Revista/Evento], [v.?, n.?, p.?-?], 2021. Disponível em: [URL]. Acesso em: 16 ago. 2025.

XAVIER, L.; SILVA, F.; CARVALHO, R.; et al. Integrating technical debt management into agile development: Lessons learned and empirical evidence. Journal of Systems and Software, v. 132, p. 123–140, 2017. Acesso em: 14 ago. 2025.

YLI-HUUMO, J.; SMOLANDER, K.; et al. Where is current research on blockchain technology? – a systematic review. PLoS ONE, v. 9, n. 10, p. e110522, 2016. Disponível em: https://www.researchgate.net/publication/308877750_Where_Is_Current_Research_on_Blockchain_Technology-A_Systematic_Review. Acesso em: 14 ago. 2025.