RESSALVA

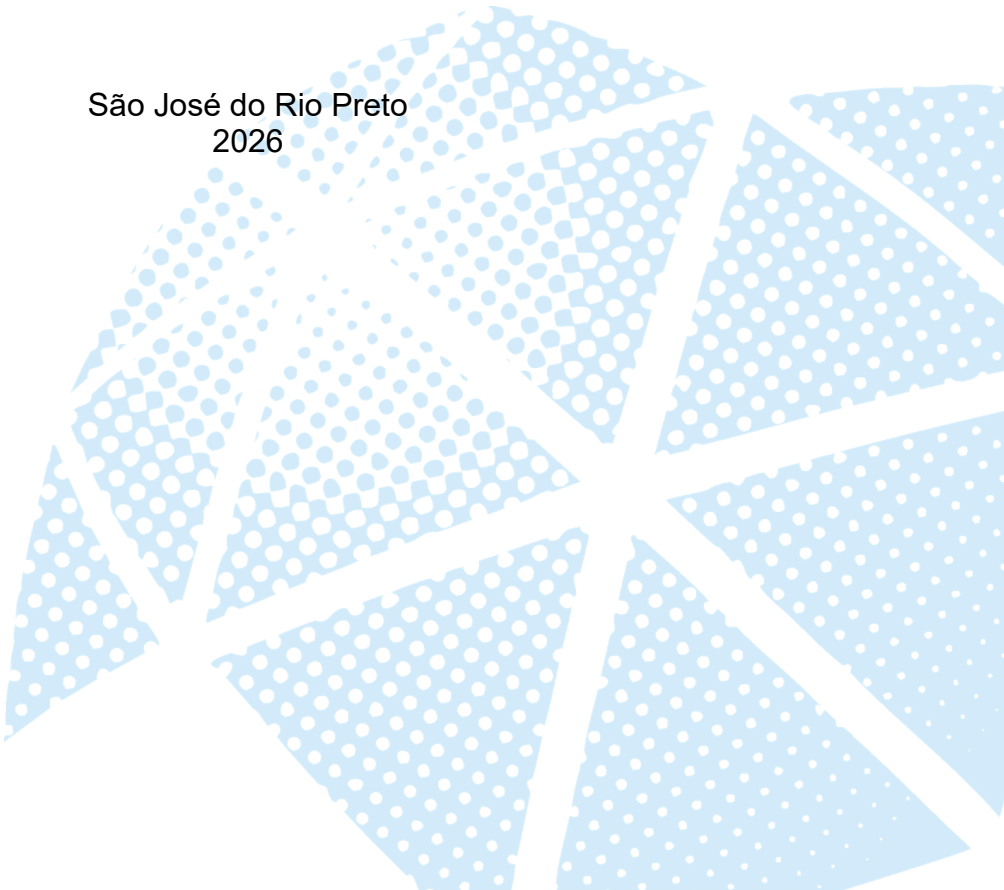
Atendendo solicitação da autora, o texto completo desta dissertação será disponibilizado somente a partir de 12/08/2028.

UNIVERSIDADE ESTADUAL PAULISTA – UNESP
Instituto de Biociências, Letras e Ciências Exatas - Campus de São José do Rio Preto

NICOLE DA SILVA CARVALHO

FRAMEWORK DE GESTÃO DO CONHECIMENTO PARA CONTROLE DE DÍVIDAS TÉCNICAS

São José do Rio Preto
2026



NICOLE DA SILVA CARVALHO

FRAMEWORK DE GESTÃO DO CONHECIMENTO PARA CONTROLE DE DÍVIDAS TÉCNICAS

Dissertação apresentada à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas - Campus de São José do Rio Preto para obtenção do título de Mestre(a) em Ciência da Computação.

Área de Concentração: Computação Aplicada

Orientador(a): Profª. Drª. Rogéria Cristiane Gratão de Souza

São José do Rio Preto
2026

C331f

Carvalho, Nicole da Silva

Framework de gestão do conhecimento para controle de dívidas técnicas / Nicole da Silva Carvalho. -- São José do Rio Preto, 2026

113 f. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientadora: Rogéria Cristiane Gratão de Souza

1. Ciência da computação. 2. Engenharia de software. 3. Framework. 4. Gestão do conhecimento. 5. Desenvolvimento de software. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

IMPACTO POTENCIAL DESTA PESQUISA

O impacto desta pesquisa está na proposição de uma abordagem para apoiar o controle de Dívidas Técnicas com Gestão do Conhecimento em projetos de software. O framework LIFT integra a captura, organização e reutilização do conhecimento aos eventos do Scrum, com apoio da base de conhecimento. Assim, preserva a memória organizacional e apoia decisões.

POTENTIAL IMPACT OF THIS RESEARCH

The impact of this research lies in proposing an approach to support Technical Debt control via Knowledge Management in software projects. The LIFT framework integrates knowledge capture, organization, and reuse into Scrum events, supported by a knowledge base. Thus, it preserves organizational memory and supports decisions.

NICOLE DA SILVA CARVALHO

FRAMEWORK DE GESTÃO DO CONHECIMENTO PARA CONTROLE DE DÍVIDAS TÉCNICAS

Dissertação apresentada à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas - Campus de São José do Rio Preto, para obtenção do título de Mestre(a) em Ciência da Computação.

Área de Concentração: Computação Aplicada

Data da defesa: 12 / 08 / 2026

Banca Examinadora:

_____
Profª. Drª. Rogéria Cristiane Gratão de Souza
UNESP - Instituto de Biociências, Letras e Ciências Exatas - Campus de São José do Rio Preto

_____
Prof. Dr. João Marcelo Rondina
FAMERP – Faculdade de Medicina de Rio Preto

_____
Prof. Dr. Danilo Medeiros Eler
UNESP– Câmpus de Presidente Prudente

Para minha família, meu cônjuge e amigos de profissão que me guiam, me ensinam e me inspiram.

AGRADECIMENTOS

Agradeço a Deus, pela saúde, pela energia e pela clareza mental que me permitiram superar os desafios desta caminhada.

Ao meu cônjuge, pelo companheirismo e amor diários. Agradeço por compreender minhas ausências, por ouvir meus desabafos e por tornar a rotina da pós-graduação muito mais leve. Esta conquista é nossa.

À minha família, que é a minha base essencial. Agradeço pelo carinho, pela torcida constante e por sempre acreditarem no valor da minha dedicação e dos meus estudos.

Aos meus amigos e colegas de profissão, pelas discussões técnicas e pela troca de experiências no ecossistema de *software*.

À minha orientadora, pelas orientações fundamentais e contribuições valiosas ao longo destes anos.

Por fim, a todos os profissionais que participaram da avaliação do framework LIFT, doando seu tempo e conhecimento prático para a validação deste trabalho.

RESUMO

A Dívida Técnica, uma metáfora criada por Ward Cunningham, descreve o custo futuro das decisões temporárias e rápidas, as quais, se não gerenciadas, podem acumular e comprometer o *software*. Este estudo aborda a importância da gestão de Dívida Técnica no desenvolvimento de *software*, enfatizando como o conhecimento agregado ao longo do ciclo de vida do projeto impacta diretamente na qualidade, escalabilidade e manutenção do sistema. Os diferentes tipos de Dívidas Técnicas podem ser "dívidas auto-admitidas", quando a equipe está ciente das concessões feitas em detrimento da qualidade, ou "dívidas inconscientes", quando surgem problemas de qualidade inesperados como consequência do acúmulo de Dívida Técnica, ambos afetando negativamente o *software*. Já é de amplo conhecimento que sistemas com altos níveis de Dívidas Técnicas demandam mais esforço de manutenção e podem até se tornar inviáveis. Com base nisso, o trabalho apresenta um *framework* para a gestão do conhecimento com o intuito de apoiar o controle dos diferentes tipos de Dívidas Técnicas, centralizando boas práticas e lições aprendidas para facilitar o acesso e uso do conhecimento técnico por todos os membros da equipe durante o ciclo de vida do *software*. Para avaliar o *framework*, foi realizado um estudo de caso guiado com 15 profissionais da área de tecnologia. Os resultados obtidos demonstraram boa aceitação do *framework* com base em quatro dimensões analíticas: clareza e compreensão, utilidade percebida, viabilidade e adoção, e impacto organizacional. Com isso, constatou-se o potencial do *framework* para contribuir com o controle e a redução do acúmulo de Dívidas Técnicas e reter o conhecimento da equipe. Contudo, os participantes sinalizaram barreiras de viabilidade prática quanto ao esforço documental manual e o receio de prolongar as cerimônias do Scrum, sugerindo a incorporação de recursos de Inteligência Artificial para automação dos registros na base de conhecimento em trabalhos futuros.

Palavras-chave: dívida técnica, gestão do conhecimento, qualidade de *software*, ciclo de desenvolvimento de *software*.

ABSTRACT

Technical Debt, a metaphor coined by Ward Cunningham, describes the future cost of temporary and rapid decisions, which, if not managed, can accumulate and compromise the software. This study addresses the importance of Technical Debt management in software development, emphasizing how the knowledge accumulated throughout the project lifecycle directly impacts the quality, scalability, and maintainability of the system. Different types of Technical Debt can be "self-acknowledged debt", when the team is aware of the trade-offs made at the expense of quality, or "unconscious debt", when unexpected quality problems arise as a consequence of the accumulation of Technical Debt, both negatively affecting the software. It is widely known that systems with high levels of Technical Debt require more maintenance effort and can even become unviable. Based on this, the work presents a framework for knowledge management with the aim of supporting the control of different types of Technical Debt, centralizing best practices and lessons learned to facilitate access to and use of technical knowledge by all team members throughout the software lifecycle. To evaluate the framework, a guided case study was conducted with 15 technology professionals. The results showed good acceptance of the framework based on four analytical dimensions: clarity and understanding, perceived usefulness, feasibility and adoption, and organizational impact. This demonstrated the framework's potential to support the organization, accessibility, and reuse of knowledge related to Technical Debt and retain team knowledge. However, participants indicated practical feasibility barriers regarding manual documentation effort and the fear of prolonging Scrum ceremonies, suggesting the incorporation of Artificial Intelligence resources for automating records on knowledge base in future work.

Keywords: technical debt; knowledge management; software quality; software development lifecycle.

LISTA DE FIGURAS

Figura 1 - Modelo de Gestão do Conhecimento.....24
Figura 2 - Frequência dos tipos de DT nos estudos voltados para identificação.....39
Figura 3 - Frequência dos tipos de DT nos estudos sobre gestão 41
Figura 4 - Estrutura do framework LIFT 48
Figura 5 - Gráfico de participantes da pesquisa.....64
Figura 6 - Gráfico de uso atual do Scrum por participantes da pesquisa 65
Figura 7 - Gráfico sobre Clareza e Compreensão do LIFT 66
Figura 8 - Gráfico sobre Utilidade Percebida do LIFT 68
Figura 9 - Gráfico sobre Viabilidade e Adoção do LIFT.....70
Figura 10 - Gráfico sobre Impacto Organizacional do LIFT.....71
Figura 11 - Gráfico sobre Avaliação dos artefatos do LIFT 72

LISTA DE TABELAS

Tabela 1 - Questões de pesquisa.....30
Tabela 2 - Critérios de inclusão e exclusão.....30
Tabela 3 - *Strings* de busca.....30
Tabela 4 - Relação de bases de dados e trabalhos que responderam Q1 e/ou Q2..32

LISTA DE ABREVIATURAS E SIGLAS

APIs *Application Programming Interfaces*

CSRF *Cross-Site Request Forgery*

DS *Docstrings*

DT Dívida Técnica

DTs Dívidas Técnicas

GC Gestão do Conhecimento

ITSM *Information Technology Service Management*

LIFT *Learn, Inspect, Fix, Track*

LLMs *Large Language Models*

PRISMA *Preferred Reporting Items for Systematic Reviews and Meta-Analyses*

RSL Revisão Sistemática da Literatura

SATD *Self-Admitted Technical Debt*

SSL *Secure Sockets Layer*

UX *User Experience*

SUMÁRIO

1 INTRODUÇÃO 15
1.1 CONTEXTUALIZAÇÃO 15
1.2 JUSTIFICATIVA 17
1.3 OBJETIVOS 18
1.4 METODOLOGIA 18
1.5 ORGANIZAÇÃO DA MONOGRAFIA 20
2 FUNDAMENTAÇÃO TEÓRICA 21
2.1 EMBASAMENTO CONCEITUAL 21
2.1.1 Tipos de Dívidas Técnicas 21
2.1.2 Modelo de Gestão do Conhecimento 23
2.1.3 Impactos da Gestão de Conhecimento para os tipos de Dívidas Técnicas 25
2.2 REVISÃO SISTEMÁTICA DA LITERATURA 29
2.3 RESULTADOS DA REVISÃO SISTEMÁTICA DA LITERATURA 31
2.3.1 Quais são as Estratégias existentes para Identificação de Dívidas Técnicas? 34
2.3.2 Quais são os Métodos para Gerenciar a Dívida Técnica? 37
2.3.3 Análise Crítica e Síntese dos Resultados 39
2.4 ABORDAGEM DE PESQUISA ADICIONAL: TÉCNICA DE BOLA DE NEVE (*SNOWBALL*) 43
2.5 CONSIDERAÇÕES FINAIS 45
3 *FRAMEWORK* LIFT 46
3.1 CONCEPÇÃO GERAL 46
3.1.1 Estruturação do *framework* 48
3.1.2 Princípios fundamentais 50
3.2 ATORES, FUNÇÕES E EVENTOS 52
3.3 BASE DE CONHECIMENTO 55
3.3.1 Artefato 1: Guia de diferentes tipos de Dívidas Técnicas 56
3.3.2 Artefato 2: *Checklist* técnico e Lições aprendidas 56
3.3.3 Artefato 3: Histórico de Dívidas Técnicas 58
3.4 CONSIDERAÇÕES FINAIS 59
4 RESULTADOS E DISCUSSÃO 61
4.1 VALIDAÇÃO DO *FRAMEWORK* LIFT 61
4.1.1 Instrumentos de Validação 61
4.1.2 Identificação e Seleção de Participantes 62
4.1.3 Estudo de Caso Guiado 62

4.2 ANÁLISE DOS RESULTADOS 64
4.3 IDENTIFICAÇÃO DE PONTOS DE MELHORIA 72
4.4 CONSIDERAÇÕES FINAIS 74
5 CONCLUSÕES 76
5.1 DISCUSSÃO GERAL 76
5.2 CONTRIBUIÇÕES 77
5.3 TRABALHOS FUTUROS 77
5.4 CONSIDERAÇÕES FINAIS 78
REFERÊNCIAS 80
APÊNDICE A: REVISÃO SISTEMÁTICA DA LITERATURA: 40 ESTUDOS SELECIONADOS ENTRE OS 112 ENCONTRADOS 89
APÊNDICE B: 16 ESTUDOS SELECIONADOS PARA APLICAÇÃO DA TÉCNICA SNOWBALL 91
APÊNDICE C: GUIA DE DIFERENTES TIPOS DE DÍVIDAS TÉCNICAS 92
APÊNDICE D: *TEMPLATES* PARA *CHECKLIST* TÉCNICO E LIÇÕES APRENDIDAS 95
APÊNDICE E: *TEMPLATE* PARA HISTÓRICO DE DÍVIDAS TÉCNICAS 98
APÊNDICE F: MANUAL TÉCNICO DESCRITIVO SOBRE O *FRAMEWORK* LIFT 99
APÊNDICE G: QUESTIONÁRIO PARA AVALIAR O *FRAMEWORK* LIFT 106
APÊNDICE H: TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO 111
DADOS CURRICULARES 113

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O desenvolvimento de *software* moderno é complexo e dinâmico. As decisões técnicas tomadas ao longo do ciclo de vida de um projeto impactam diretamente a qualidade, a escalabilidade e a capacidade de manutenção do produto final. Um dos conceitos fundamentais para compreender e gerir esses impactos é o da Dívida Técnica (DT), um termo introduzido por Ward Cunningham para descrever o custo futuro das decisões tomadas rapidamente e das soluções provisórias implementadas durante o ciclo de desenvolvimento (Cunningham, 1992). A metáfora da Dívida Técnica permite visualizar como as decisões realizadas para cumprir os prazos ou demandas do mercado podem, ao longo do tempo, comprometer o sucesso do projeto, pois a "dívida" acumula e precisa em algum momento ser "paga" e, muitas vezes, este pagamento demanda trabalho adicional, necessidade de reestruturação, refatoração e/ou correções emergentes para garantir a qualidade do *software* ao cliente final.

A Dívida Técnica pode se manifestar de diversas maneiras, sendo comumente classificada em duas categorias principais: a "dívida técnica auto-admitida", que ocorre quando a equipe de desenvolvimento está ciente das decisões para acelerar o processo de desenvolvimento e alcançar prazos mais curtos; e a "dívida técnica inconsciente", que se forma quando problemas de qualidade se acumulam de maneira não intencional e frequentemente passam despercebidos (Rios et al., 2020). Essas categorias de DT impactam a qualidade do *software* de modos distintos, mas igualmente prejudiciais. Como observa Lim et al. (2012), sistemas com altos níveis de Dívida Técnica tendem a apresentar mais defeitos e exigem maiores esforços de manutenção, o que, em casos extremos, pode inviabilizar a evolução do *software*.

Embora a literatura reconheça a existência tanto da dívida técnica auto-admitida quanto da inconsciente, estudos e experiências práticas indicam que a dívida técnica inconsciente tende a ser a mais frequente em ambientes reais de desenvolvimento de *software*. Isso ocorre, principalmente, devido à limitação de recursos, à rotatividade de equipes, à ausência de documentação adequada e à falta de conhecimento aprofundado sobre o sistema, fatores comuns em projetos legados.

Nesses cenários, decisões técnicas inadequadas ou subótimas são frequentemente tomadas sem a plena consciência de seus impactos futuros, resultando no acúmulo gradual e não intencional de Dívida Técnica. Diferentemente da dívida auto-admitida, que é assumida de forma consciente para atender prazos ou restrições específicas, a dívida inconsciente é mais difícil de ser identificada e gerenciada, tornando-se um desafio significativo para a manutenção e evolução do *software*.

Além das categorias descritas, a literatura especializada sobre Dívida Técnica identifica ainda uma diversidade de tipos de DT, cada qual associado a diferentes etapas e artefatos do processo de desenvolvimento de *software*. Esses tipos incluem dívidas de arquitetura, código, documentação, testes, defeitos, infraestrutura, processos, entre outros, que possuem impactos variados sobre o desempenho e qualidade do *software*. Em sua pesquisa, Rios, Neto e Spínola (2018) mapearam quinze tipos distintos de DT, cada um afetando de modo particular a integridade do *software*, evidenciando a complexidade de se manter o equilíbrio entre velocidade de entrega e qualidade a longo prazo.

Paralelamente, a Gestão do Conhecimento (GC), surge como uma ferramenta estratégica na gestão da DT. GC envolve processos sistemáticos de criação, captura, compartilhamento e aplicação de conhecimento dentro de uma organização com o objetivo de aumentar a eficiência (O'Connor; Basri, 2014). No setor de desenvolvimento de *software*, a gestão eficaz do conhecimento é essencial para garantir que a experiência acumulada e as melhores práticas sejam documentadas e transmitidas adequadamente, evitando assim que a falta de comunicação ou de informações relevantes contribua para a criação de Dívidas Técnicas. A ausência de boas práticas de GC pode, de fato, levar ao acúmulo de DT, uma vez que em geral decisões são tomadas sem a visibilidade completa dos impactos futuros delas. Dessa maneira, a implantação de práticas de GC visa gerenciar esses riscos, assegurando que o conhecimento organizacional esteja acessível e seja utilizado de modo eficaz por toda a equipe de desenvolvimento de *software*.

Diante deste contexto, este estudo investiga como a definição de um *framework*[1] de Gerenciamento de Conhecimento pode contribuir para a centralização

[1] *Framework*: um conjunto de diretrizes, papéis e boas práticas que ajudam equipes a trabalhar de forma mais organizada, produtiva e eficiente.

e gestão de informações sobre DT, de modo a promover uma abordagem mais eficaz para o gerenciamento de seus impactos e promover a redução do acúmulo dessas dívidas no *software*. A proposta deste estudo é desenvolver um *framework* que não apenas centralize o conhecimento relacionado aos tipos de Dívidas Técnicas, mas também facilite o uso dele pela equipe, com fins de apoiar a melhoria contínua no desenvolvimento de *software*. Dessa maneira, o *framework* visa contribuir para que lições aprendidas ao lidar com Dívidas Técnicas e melhores práticas sejam acessíveis para quaisquer membros da equipe, impactando positivamente a qualidade do *software*.

1.2 JUSTIFICATIVA

A gestão dos tipos de Dívidas Técnicas possui relevância no desenvolvimento de *software*, pois as decisões tomadas ao longo do ciclo de vida do projeto impactam diretamente sua qualidade, escalabilidade e manutenibilidade (Cunningham, 1992). Atualmente é comum equipes de desenvolvimento que priorizam entregas rápidas devido à pressão de prazos e demandas de mercado, levando ao acúmulo de vários tipos de Dívidas Técnicas que afetam negativamente a qualidade do *software* a longo prazo (Rios, Neto e Spínola, 2018). Nesse contexto, práticas eficazes para gerenciar, monitorar e pagar os vários tipos de Dívidas Técnicas, tornam-se essenciais para garantir a sustentabilidade e competitividade dos *softwares* desenvolvidos.

A ausência de um meio que centralize e possibilite a captura, organização e compartilhamento do conhecimento acumulado sobre os tipos de Dívidas Técnicas gera uma série de desafios. A fragmentação de informações e a falta de documentação de boas práticas dificultam a compreensão e o gerenciamento dos tipos de Dívidas Técnicas, especialmente para equipes que lidam com alta rotatividade de membros. O Gerenciamento do Conhecimento oferece uma abordagem estruturada, possibilitando que o conhecimento sobre Dívidas Técnicas, lições aprendidas e melhores práticas sejam centralizados, acessíveis e aplicáveis de maneira contínua (O'Connor; Basri, 2014).

Além disso, a gestão sistemática e documentada dos tipos de Dívidas Técnicas possibilita que empresas de *software* estabeleçam uma governança mais eficaz sobre seus processos de desenvolvimento. Logo, a definição de um *framework* como apresentado neste trabalho poderá contribuir para o controle e a prevenção do

acúmulo dos diferentes tipos de Dívidas Técnicas, bem como facilitar a comunicação entre os membros da equipe, aumentar a eficiência do trabalho colaborativo e melhorar a qualidade final dos produtos de *software*.

Este estudo justifica-se pela necessidade de contribuir para a gestão de diferentes tipos de Dívidas Técnicas. Assim, é criado um *framework* de Gerenciamento de Conhecimento específico para apoiar a gestão dos diferentes tipos de Dívidas Técnicas, com objetivo de consolidar as melhores práticas, facilitar o compartilhamento de lições aprendidas e incentivar uma abordagem proativa para minimizar os impactos negativos dos tipos de Dívidas Técnicas não gerenciadas no ciclo de vida do *software*.

1.3 OBJETIVOS

Como objetivo geral, tem-se o desenvolvimento de um mecanismo de gestão baseado em Gerenciamento do Conhecimento para apoiar o monitoramento e controle dos diferentes tipos de Dívidas Técnicas, por meio da organização, do registro e da centralização das informações relacionadas a essas dívidas. Neste contexto, foi definida a seguinte questão norteadora para o presente trabalho: “Como organizar e disponibilizar conhecimento sobre Dívidas Técnicas?”.

Assim, como objetivo específico, foi estruturado um *framework* com o intuito de fomentar a documentação e a reutilização do conhecimento existente sobre Dívidas Técnicas, tais como lições aprendidas e boas práticas previamente identificadas pelas equipes de desenvolvimento de *software*. O *framework* não tem como propósito gerar ou impor tais conhecimentos, mas contribuir para que eles estejam estruturados, acessíveis e reutilizáveis, de forma a embasar as decisões técnicas voltadas para evitar o acúmulo de Dívidas Técnicas, bem como sensibilizar as equipes sobre a importância de controlar tais dívidas.

1.4 METODOLOGIA

A metodologia adotada para o desenvolvimento do *framework* foi dividida em quatro etapas principais.

A primeira considerou a revisão da literatura relacionada a Dívidas Técnicas e Gerenciamento do Conhecimento, com o intuito de: analisar os conceitos existentes

na literatura sobre o que são Dívidas Técnicas, suas categorias e implicações nos projetos de desenvolvimento de *software*; investigar as práticas recomendadas para a prevenção do acúmulo, o monitoramento e o pagamento de Dívidas Técnicas, bem como para o reuso de lições aprendidas de casos de Dívidas Técnicas anteriores; e revisar estudos que abordam diferentes metodologias e ferramentas utilizadas para gerenciar os tipos de Dívidas Técnicas em projetos de desenvolvimento de *software* para entender como as equipes têm abordado o tema e quais ferramentas têm se mostrado eficazes na gestão dessas dívidas.

Na sequência, durante a segunda etapa, houve o refinamento das observações feitas a partir da análise dos estudos revisados, com o intuito de: realizar uma síntese das melhores práticas relacionadas à gestão dos tipos de Dívidas Técnicas, destacando quais estratégias são mais eficazes para a prevenção do acúmulo de Dívidas Técnicas, monitoramento e pagamento de dívidas, assim como a importância de documentar lições aprendidas; e identificar os desafios recorrentes enfrentados pelas equipes na gestão dos tipos de Dívidas Técnicas, permitindo que o *framework* aborde especificamente essas questões.

Já a terceira etapa considerou a construção do *framework* a partir das observações formuladas e das melhores práticas identificadas. Para tanto, o *framework* foi estruturado com o intuito de abordar diferentes aspectos da gestão dos tipos de Dívidas Técnicas.

Finalmente, a última etapa foi a validação do *framework* por meio de um estudo de caso guiado e da posterior aplicação de questionário com profissionais que participam de projetos reais. O estudo de caso foi realizado em um ambiente de simulação e, para tanto, o LIFT foi estruturado com dados fictícios em uma área de trabalho compartilhada usando o Confluence. Essa etapa possibilitou a obtenção de uma avaliação tanto quantitativa, por meio da análise estatística das respostas fechadas do questionário, quanto qualitativa, a partir das percepções, sugestões e críticas obtidas durante o estudo de caso guiado e das respostas abertas do questionário. Esses dados permitiram verificar o grau de aplicabilidade do *framework* na prática, identificar pontos fortes e áreas que necessitam de ajustes, além de orientar melhorias com base em evidências concretas.

1.5 ORGANIZAÇÃO DA MONOGRAFIA

Esta monografia está estruturada em cinco capítulos para oferecer uma visão do desenvolvimento do trabalho, conforme segue.

No Capítulo 1 é feita uma introdução para embasar este estudo. Neste capítulo há a introdução geral do tema, destacando sua relevância, justificativa para realização do estudo e destaque do objetivo.

O Capítulo 2 possui foco na contextualização teórica dos conceitos centrais deste trabalho. É neste capítulo que há a definição de dívida técnica, dos diferentes tipos de dívidas técnicas, da gestão do conhecimento e, adicionalmente, o capítulo aborda a revisão sistemática da literatura que foi realizada para identificar as considerações relevantes acerca de trabalhos selecionados sobre este campo de estudo na engenharia de *software*.

No Capítulo 3 são apresentados os resultados obtidos ao longo da realização deste estudo, culminando na definição do *framework* LIFT.

No Capítulo 4 há foco nos resultados e discussão após validação do *framework* LIFT e aplicação de questionário aos participantes da pesquisa.

O Capítulo 5, por fim, concentra-se nas conclusões e trabalhos futuros.

5 CONCLUSÕES

5.1 DISCUSSÃO GERAL

O acúmulo de Dívidas Técnicas (DTs) representa um dos desafios mais complexos e persistentes no desenvolvimento moderno de *software*, impactando de forma direta a sustentabilidade, a escalabilidade e a manutenção dos sistemas a longo prazo. Como evidenciado na Revisão Sistemática da Literatura conduzida neste trabalho, a indústria de *software* tende a gerenciar e identificar majoritariamente os passivos que são facilmente mensuráveis por ferramentas automáticas, com forte dominância das dívidas técnicas de código e arquitetura, deixando lacunas críticas em outros tipos de dívidas técnicas, como as DTs de processo, serviços, requisitos e versionamento, por exemplo. Diante desse cenário, esta pesquisa investigou a interseção da Gestão do Conhecimento para apoiar o monitoramento e controle de diferentes tipos de Dívidas Técnicas, resultando no desenvolvimento do *framework* LIFT, o qual integra o Ciclo de Gestão do Conhecimento de Dalkir (2005) aos eventos da metodologia ágil Scrum.

O LIFT demonstrou ser uma abordagem inovadora ao transformar as ocorrências de Dívida Técnica em gatilhos estratégicos para a conversão de conhecimento tácito em explícito, centralizados em uma base de conhecimento composta por três artefatos interdependentes: o Guia de tipos de DT, o *Checklist* técnico e Lições aprendidas, e o Histórico de DTs.

A validação do *framework*, realizada por meio de um estudo de caso guiado com 15 profissionais do mercado de Tecnologia da Informação, confirmou sua relevância prática e seu rigor conceitual. Os resultados quantitativos e qualitativos indicaram boa aceitação das equipes com base nas quatro dimensões analíticas avaliadas: clareza e compreensão, utilidade percebida, viabilidade e adoção, além de impacto organizacional. Paralelamente, o processo de validação expôs os desafios práticos de adoção em ambientes corporativos de ritmo acelerado, evidenciando o atrito natural entre o esforço documental exigido para manter a base de conhecimento ativa e as pressões comerciais por entregas rápidas. Conclui-se, portanto, que o LIFT propicia a evolução da maturidade técnica de equipes ágeis ao promover a documentação, o acesso e a reutilização do conhecimento sobre Dívidas Técnicas durante um processo de desenvolvimento de *software*, contribuindo para uma gestão mais eficaz.

5.2 CONTRIBUIÇÕES

As contribuições geradas por esta dissertação estendem-se tanto ao ecossistema acadêmico quanto à prática profissional em Tecnologia da Informação, mais especificamente na Engenharia de *Software*, articulando-se nos seguintes pontos:

- *Framework* LIFT: Concepção de um *framework* original e estruturado que integra práticas de Gestão do Conhecimento diretamente à rotina operacional dos eventos Scrum, eliminando a fragmentação documental e fornecendo papéis claros aos atores ágeis (*Scrum Master*, *Product Owner* e Equipe de Desenvolvimento) no manejo de diferentes tipos de Dívidas Técnicas;
- Base de Conhecimento reutilizável: Criação de três artefatos operacionais e padronizados - o Guia de tipos de DT, o *Checklist* técnico e Lições aprendidas, e o Histórico de DTs, que atuam como um ativo organizacional, reduzindo a dependência de especialistas e a perda de conhecimento técnico causada pela rotatividade de equipes;
- Mapeamento e Síntese da literatura: Realização de uma Revisão Sistemática da Literatura baseada no protocolo PRISMA (40 estudos analisados) e complementada pela técnica de *Snowballing* (6 estudos adicionais), resultando em uma análise crítica sobre a frequência e as lacunas existentes nos métodos atuais de identificação e gestão dos 15 tipos de Dívidas Técnicas catalogados na literatura;
- Contribuição à Governança Ágil: Mapeamento das percepções de profissionais de mercado, revelando o atrito cultural entre o esforço para documentação técnica e as prioridades comerciais das organizações de *software*, pavimentando novas oportunidades para evolução da área.

5.3 TRABALHOS FUTUROS

A partir das limitações identificadas durante o estudo de caso guiado e dos pontos de melhoria sugeridos pelos profissionais que participaram da validação, propõem-se as seguintes frentes para trabalhos futuros:

- Automação e integração do *framework* LIFT com Inteligência Artificial (IA): Investigar o uso de Grandes Modelos de Linguagem (LLMs) e agentes inteligentes integrados ao pipeline de desenvolvimento e ferramentas de

ITSM, como Jira. O intuito é automatizar o preenchimento de lições aprendidas a partir de *commits*, além do rastreamento automático de DTs geradas por assistentes virtuais de código, diminuindo a sobrecarga documental manual apontada na validação;

- Abordagem de registro contínuo assíncrono: Evoluir a dinâmica operacional do LIFT para permitir e incentivar o registro assíncrono e imediato das dívidas técnicas no decorrer das atividades diárias dos desenvolvedores. O intuito é reduzir o risco de perda de detalhes críticos e otimizar o tempo dedicado exclusivamente a isso dentro das cerimônias formais do Scrum;
- Mecanismos de blindagem de velocidade e contratos de priorização: Desenvolver e testar modelos de "contratos de priorização" e regras de governança que determinem a reserva fixa de uma porcentagem da velocidade da *Sprint* (ex: 10% a 20%) voltada ao pagamento de dívidas técnicas mapeadas na base de conhecimento. O intuito é auxiliar o *Product Owner* a balizar tecnicamente o retorno do investimento temporal;
- Expansão e atualização taxonômica: Ampliar o Guia de Tipos de Dívidas Técnicas do *framework* para abranger novas categorias emergentes e essenciais no cenário de desenvolvimento moderno, incluindo Dívida de Segurança da Informação;
- Validação em ambiente de produção real: Aplicar o *framework* LIFT de forma longitudinal em um estudo de caso em ambiente real, acompanhando múltiplos ciclos de *Sprints* consecutivos em uma organização de *software* para mensurar quantitativamente o impacto do *framework* na redução real do retrabalho e na taxa de entrega de valor ao longo do tempo.

5.4 CONSIDERAÇÕES FINAIS

As reflexões ao final deste trabalho permitem afirmar que o objetivo geral da pesquisa foi alcançado. O desenvolvimento do *framework* LIFT conseguiu preencher uma lacuna relevante na área de Engenharia de *Software* ao unir os conceitos de Gestão do Conhecimento com a rotina de equipes que utilizam o Scrum. Embora os participantes tenham reconhecido o potencial da documentação sistemática de lições aprendidas e da consulta ao histórico de Dívidas Técnicas, esta pesquisa não mensurou diretamente a taxa de reincidência envolvendo DTs antes e depois da

utilização do LIFT. Dessa forma, a redução efetiva da reincidência permanece como hipótese a ser investigada em estudos longitudinais realizados em ambientes reais de desenvolvimento. Em resumo, o *framework* mostrou que evitar o acúmulo de Dívidas Técnicas não depende de regras rígidas, mas sim de criar o hábito de registrar e compartilhar o conhecimento na equipe.

A importância prática da proposta foi confirmada pelos profissionais de Tecnologia da Informação que participaram do estudo de caso. A grande concordância deles em relação às quatro dimensões analíticas avaliadas (clareza e compreensão, utilidade percebida, viabilidade e adoção, além de impacto organizacional) do LIFT na organização mostrou que o mercado precisa de ferramentas que vão além do monitoramento automático de Dívidas Técnicas de código. Os profissionais valorizam estratégias que ajudem a guardar o conhecimento técnico e evitem que informações fiquem restritas a poucas pessoas. Além disso, as críticas e alertas trazidos pelos participantes, como o medo de que o preenchimento manual dos relatórios gerasse burocracia, atrasasse as reuniões ou tirasse a agilidade do Scrum, trouxeram um realismo necessário para a pesquisa. Esses pontos não tiram o valor do *framework*, mas ajudam a definir seus limites e mostram que o uso de Inteligência Artificial para automatizar esses registros na base de conhecimento do LIFT é o caminho necessário para um melhor alinhamento do *framework* ao dia a dia ágil das empresas.

É importante destacar que, devido à necessidade de um período muito mais longo para acompanhar o *framework* operando no dia a dia de uma empresa, a validação desta pesquisa limitou-se a um contexto simulado apoiado na experiência prática dos profissionais convidados. Por essa razão, surge a necessidade evidente de validar o LIFT em cenários reais de produção no futuro. Testar o *framework* em projetos ativos e de longo prazo permitirá observar como ele se comporta diante da rotina real das equipes de engenharia de *software* ao longo de várias *Sprints* sucessivas.

Por fim, este estudo traz contribuições importantes para as pesquisas sobre processos de *software*. Ao analisar detalhadamente as características de 15 tipos de Dívidas Técnicas e propor uma forma de gerenciá-las, esta dissertação serve como base para futuros trabalhos que queiram estudar dívidas técnicas mais difíceis de medir. O LIFT se consolida como uma ferramenta madura para apoiar o controle da Dívida Técnica, evidenciando que é possível equilibrar a pressa das entregas exigidas pelo negócio com a manutenção da qualidade do *software*, desde que a empresa saiba guardar e usar a sua própria memória sobre Dívidas Técnicas.

REFERÊNCIAS

ALDAEEJ, A.; SEAMAN, C. A rule-based Decision Model to Support Technical Debt decisions: a Multiple Case Study of Web and Mobile App Startups. **Information and Software Technology**, v. 175, p. 107542, nov. 2024.

ALHUSAIN, S. Predicting Relative Thresholds for Object Oriented Metrics. **ArXiv (Cornell University)**, 1 jan. 2021.

ALOMAR, E. A.; CHRISTIANS, B.; BUSHO, M.; ALKHALID, A. H.; OUNI, A.; NEWMAN, C.; MKAOUER, M. W. SATDBailiff-mining and Tracking self-admitted Technical Debt. **Science of Computer Programming**, v. 213, p. 102693, jan. 2022.

AMPATZOGLOU, A.; CHATZIGEORGIOU, A.; ARVANITOU, E. M.; BIBI, S. SDK4ED: a Platform for Technical Debt Management. **Software: Practice and Experience**, v. 52, n. 8, p. 1879–1902, 4 maio 2022. Acesso em: 26 fev. 2025.

ANN , R. L.; OUZZIF, Z. Technical Debt in Hardware Systems and Elements. **2021 IEEE Aerospace Conference**. 6 mar. 2021.

APA, C.; JERONIMO, H.; NASCIMENTO, L. M.; VALLESPIR, D.; TRAVASSOS, G. H. The Perception and Management of Technical Debt in Software Startups. **Fundamentals of Software Startups**, p. 61–78, 1 jan. 2020a.

APA, C.; SOLARI, M.; VALLESPIR, D.; TRAVASSOS, G, H. A Taste of the Software Industry Perception of Technical Debt and its Management in Uruguay: A survey in software industry. **ESEM '20: Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**. 5 out. 2020b.

ARDIMENTO, P.; AVERSANO, L.; BERNARDI, M. L.; CIMITILE, M.; IAMMARINO, M. Using Deep Temporal Convolutional Networks to just-in-time Forecast Technical Debt Principal. **Journal of Systems and Software**, v. 194, p. 111481, dez. 2022.

BATISTA, F. F. Modelo de gestão do conhecimento para a administração pública brasileira: como implementar a gestão do conhecimento para produzir resultados em benefício do cidadão. Acesso em: 5 maio. 2025.

BESKER, T.; GHANBARI, H.; MARTINI, A.; BOSCH, J. The Influence of Technical Debt on Software Developer Morale. **Journal of Systems and Software**, v. 167, p. 110586, 1 set. 2020.

BESKER, T.; MARTINI, A.; BOSCH, J. The use of incentives to promote technical debt management. **Information and Software Technology**, v. 142, p. 106740, fev. 2022.

BONFIM, V.; BENITTI, F. Requirements debt: causes, consequences, and Mitigating Practices. **International Conferences on Software Engineering and Knowledge Engineering**, v. 2022, p. 13–18, 1 jul. 2022.

BOROWA, K.; ZALEWSKI, A.; SACZKO, A. Living with Technical Debt - A Perspective from the Video Game Industry. **IEEE Software**, 2021.

BROWN, N.; OZKAYA, I.; SANGWAN, R.; SEAMAN, C.; SULLIVAN, K.; ZAZWORKA, N.; CAI, Y.; GUO, Y.; KAZMAN, R.; KIM, M.; KRUCHTEN, P.; LIM, E.; MACCORMACK, A.; NORD, R. Managing Technical Debt in software-reliant Systems. **Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research - FoSER '10**, 2010.

BUKOWITZ, W. R.; WILLIAMS, R. L.; ALBERTO, C. Manual de Gestão do Conhecimento : Ferramentas e Técnicas que criam valor para a Empresa. Porto Alegre: **Bookman**, 2002.

CAGLAYAN, D.; ÖZCAN-TOP, Ö. Revisiting Technical Debt Types and Indicators for Software Systems. **SAC '24: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing** 8 abr. 2024.

CERVANTES, H.; KAZMAN, R. Software archinaut: a Tool to Understand architecture, Identify Technical Debt Hotspots and Manage Evolution. **IEEE/ACM International Conference on Technical Debt (TechDebt).** 28 jun. 2020.

CHOO, C. W.; BONTIS, N. The Strategic Management of Intellectual Capital and Organizational Knowledge. New York: **Oxford University Press**, 2002.

COSTA, W. P. L. B. da; SILVA, J. D. da ; BRITO, L. M. P.; SILVA, S. L. Modelo de Gestão do Conhecimento. **Revista de Administração, Sociedade e Inovação**, v. 9, n. 2, p. 46–68, 1 maio 2023. Acesso em: 3 maio. 2025.

CRESPO, Y.; LÓPEZ-NOZAL, C.; MARTICORENA-SÁNCHEZ, R.; GONZALO-TASIS, M.; PIATTINI, M. The Role of Awareness and Gamification on Technical Debt Management. **Information and Software Technology**, v. 150, p. 106946, out. 2022.

CUNNINGHAM, W. **The WyCash Portfolio Management System. 1992.** Disponível em: https://c2.com/doc/oopsla92.html. Acesso em: 2 set. 2024.

DALKIR, K. Knowledge Management in Theory and Practice. Amsterdam ; Boston: **Elsevier/Butterworth Heinemann**, 2005.

DAS, D.; MARUF, A. A.; ISLAM, R.; LAMBARIA, N.; KIM, S.; ABDELFATTAH, A. S.; CERNY, T.; FRAJTAK, K.; BURES, M.; TISNOVSKY, P. Technical Debt Resulting from Architectural Degradation and Code Smells: a systematic mapping study. **ACM SIGAPP Applied Computing Review**, v. 21, n. 4, p. 20–36, dez. 2021.

DAVENPORT, L.; PRUSAK, T. H. Conhecimento empresarial: Como as organizações gerenciam o seu capital intelectual. **Alta Books**, 1998.

FARIAS, M. A. de F.; NETO, M. G. de M.; KALINOWSKI, M.; SPÍNOLA, R. O. Identifying self-admitted technical debt through code comment analysis with a contextualized vocabulary. **Information and Software Technology**, v. 121, p. 106270, maio 2020.

FOWLER, M.; BECK, K. Refactoring: Improving the Design of Existing Code. Boston: **Addison-Wesley**, 2019.

FRAGOSO, E.; VILLA, L. F.; GIRALDO, M. Strategies, Information Technologies and Models for Knowledge management in Software Development Companies: a Systematic Review of the Literature. **Communications in Computer and Information Science**, p. 388–398, 1 jan. 2023.

FREIRE, S.; RIOS, N.; MENDONÇA, M.; FALESSI, D.; SEAMAN, C.; IZURIETA, C.; SPÍNOLA, R. O. Actions and impediments for technical debt prevention: results from a global family of industrial surveys. **Proceedings of the 35th Annual ACM Symposium on Applied Computing**, 29 mar. 2020.

GAMA, E.; FREIRE, S.; MENDONÇA, M.; SPÍNOLA, R. O.; PAIXAO, M.; CORTÉS, M. I. Using Stack Overflow to Assess Technical Debt Identification on Software Projects. **Proceedings of the XXXIV Brazilian Symposium on Software Engineering**, p. 730–739, 21 out. 2020.

GOMES, R. A.; BARBOSA, L.; SUZANA, R. Anticipating Identification of Technical Debt Items in Model-Driven Software Projects. **SBES '20: Proceedings of the XXXIV Brazilian Symposium on Software Engineering.** v. 5, p. 740–749, 21 out. 2020.

HOWARD, K. **Avoiding Technical Bankruptcy in System development: a Process to Reduce the Risk of Accumulating Technical Debt**. Disponível em: https://mountainscholar.org/items/989fd093-b064-4503-85ea-856eb7396e72. Acesso em: 13 set. 2024.

JUNIOR, H. J.; TRAVASSOS, G. H. Consolidating a Common Perspective on Technical Debt and Its Management through a Tertiary Study. **Information and Software Technology**, v. 149, p. 106964, set. 2022.

KLEINWAKS, H.; BATCHELOR, A.; BRADLEY, T. An Empirical Survey on the Prevalence of Technical Debt in Systems Engineering. **INCOSE International Symposium**, v. 33, n. 1, p. 1640–1658, 1 jul. 2023a.

KLEINWAKS, H.; BATCHELOR, A.; BRADLEY, T. H. Technical debt in systems engineering—A systematic literature review. **Systems Engineering**, 10 abr. 2023b.

KOKOL, P.; ZAGORANSKI, S.; KOKOL, M. Software Development with Scrum: a Bibliometric Analysis and Profile. **Library Philosophy and Practice (e-journal)**. Acesso em: 27 jun. 2025.

KOZANIDIS, N.; VERDECCHIA, R.; GUZMAN, E. Asking about Technical Debt: Characteristics and Automatic Identification of Technical Debt Questions on Stack Overflow. **ESEM '22: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**. 7 set. 2022.

KRUCHTEN, P.; NORD, R. L.; OZKAYA, I. Technical Debt: from Metaphor to Theory and Practice. **IEEE Software**, v. 29, n. 6, p. 18–21, nov. 2012.

KRUCHTEN, P.; NORD, R.; IPEK OZKAYA. Managing Technical Debt: Reducing Friction in Software Development. **Addison-Wesley Professional**, 2019.

LIM, E.; TAKSANDE, N.; SEAMAN, C. A Balancing Act: What Software Practitioners Have to Say about Technical Debt. **IEEE Software**, v. 29, n. 6, p. 22–27, 23 ago. 2012.

MAIKANTIS, T.; TSINTZIRA, A.-A.; AMPATZOGLOU, A.; ARVANITOU, E.-M.; CHATZIGEORGIOU, A.; STAMELOS, I.; BIBI, S.; DELIGIANNIS, I. Software Architecture Reconstruction via a Genetic Algorithm: Applying the Move Class Refactoring. **24th Pan-Hellenic Conference on Informatics**, p. 135–139, 20 nov. 2020.

MANDIĆ, V.; TAUŠAN, N.; RAMAČ, R. The Prevalence of the Technical Debt Concept in Serbian IT Industry: results of a national-wide survey. **IEEE/ACM International Conference on Technical Debt (TechDebt)**. 28 jun. 2020.

MELO, A.; FAGUNDES, R.; LENARDUZZI, V.; SANTOS, W. B. Identification and Measurement of Requirements Technical Debt in Software development: a Systematic Literature Review. **Journal of Systems and Software**, v. 194, p. 111483, dez. 2022.

MENS, T.; TOURWE, T. A Survey of Software Refactoring. **IEEE Transactions on Software Engineering**, v. 30, n. 2, p. 126–139, fev. 2004.

MOLNAR, A.-J.; MOTOGNA, S. Long-Term Evaluation of Technical Debt in Open-Source Software. **ArXiv (Cornell University)**, 5 out. 2020.

MORESCHINI, S.; LENARDUZZI, V.; COBA, L. Towards a Technical Debt for AI-based Recommender System. **TechDebt '24: Proceedings of the 7th ACM/IEEE International Conference on Technical Debt** v. 6, p. 36–39, 14 abr. 2024.

NIELSEN, M. E.; MADSEN, C. Ø. Stakeholder Influence on Technical Debt Management in the Public sector: an Embedded Case Study. **Government Information Quarterly**, p. 101706, abr. 2022.

NONAKA, I. The Knowledge-Creating Company. Disponível em: https://lumsa.it/sites/default/files/UTENTI/u95/LM51_ITA_The%20Knowledge-Creating%20Company.pdf. Acesso em: 5 maio. 2025.

O'CONNOR, R. V.; BASRI, S. Understanding the Role of *Knowledge management* in Software Development. **International Journal of Systems and Service-Oriented Engineering**, v. 4, n. 1, p. 39–52, jan. 2014.

OBRIEN, D.; BISWAS, S.; IMTIAZ, S. M.; ABDALKAREEM, R.; SHIHAB, E.; RAJAN, H. Are Prompt Engineering and TODO Comments Friends or Foes? an Evaluation on GitHub Copilot. **ICSE '24: Proceedings of the IEEE/ACM 46th International Conference on Software Engineerin.** 12 abr. 2024.

OBRIEN, D.; BISWAS, S.; MOHAMMAD IMTIAZ, S.; ABDALKAREEM, R.; SHIHAB, E.; RAJAN, H. 23 Shades of Self-Admitted Technical Debt: an Empirical Study on Machine Learning Software. **ESEC/FSE 2022: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering**. 9 nov. 2022.

OPENJA, M.; MOROVATI, M, M.; AN, L.; KHOMH, F; ABIDI, M. Technical Debts and Faults in open-source Quantum Software systems: an Empirical Study. **Journal of Systems and Software**, v. 193, p. 111458–111458, 30 jul. 2022.

PAGE, M. J.; MCKENZIE, J. E.; BOSSUYT, P. M.; BOUTRON, I.; HOFFMANN, T. C.; MULROW, C. D.; SHAMSEER, L.; TETZLAFF, J. M.; AKL, E. A.; BRENNAN, S. E.; CHOU, R.; GLANVILLE, J.; GRIMSHAW, J. M.; HRÓBJARTSSON, A.; LALU, M. M.; LI, T.; LODER, E. W.; MAYO-WILSON, E.; MCDONALD, S.; MCGUINNESS, L. A. A Declaração PRISMA 2020: Diretriz Atualizada Para Relatar Revisões Sistemáticas. **Revista Panamericana De Salud Pública**, v. 46, p. 1, 30 dez. 2022.

PINA, D.; GOLDMAN, A.; TONIN, G. Technical Debt Prioritization: Taxonomy, Methods Results, and Practical Characteristics. **47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. 1 set. 2021.

RAMAČ, R.; MANDIĆ, V.; TAUŠAN, N.; RIOS, N.; FREIRE, S.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; PACHECO, A.; LOPEZ, G.; IZURIETA, C.; SEAMAN, C.; SPINOLA, R. Prevalence, Common Causes and Effects of Technical debt: Results from a Family of Surveys with the IT Industry. **Journal of Systems and Software**, v. 184, p. 111114, fev. 2022.

RECUPITO, G.; PECORELLI, F.; CATOLINO, G.; LENARDUZZI, V.; TAIBI, D.; DI NUCCI, D.; PALOMBA, F. Technical Debt in AI-enabled systems: on the prevalence, severity, impact, and Management Strategies for Code and Architecture. **Journal of Systems and software**, p. 112151–112151, 1 jul. 2024.

RIOS, N.; SPÍNOLA, R. O.; MENDONÇA, M.; SEAMAN, C. The Practitioners' Point of View on the Concept of Technical Debt and Its Causes and consequences: a Design for a Global Family of Industrial Surveys and Its First Results from Brazil. **Empirical Software Engineering**, v. 25, n. 5, p. 3216–3287, 13 jun. 2020.

RIOS, N.; NETO, M. G. DE M.; SPÍNOLA, R. O. 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, out. 2018.

SANTOS, C. A. D. S. Modelo de gestão do conhecimento para organizações de educação profissional e tecnológica: A comunidade de prática na implementação de um repositório digital institucional. Disponível em: https://editora.ifsp.edu.br/edifsp/catalog/view/12/12/49. Acesso em: 3 mai. 2025.

SARAIVA, D.; NETO, J. G.; KULESZA, U.; FREITAS, G.; REBOUÇAS, R.; COELHO, R. Technical Debt Tools: A Systematic Mapping Study. **Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021)** - Volume 2, pages 88-98. 1 jan. 2021.

SCHWABER, K.; SUTHERLAND, J. O Guia Definitivo para o Scrum: As Regras do Jogo. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf. Acesso em: 27 jun. 2025.

SOUEID, M. I.; MARTINS, A. F. C. Scrum and Agility beyond it: Evidences in the Brazilian Mining Industry. **Revista de Gestão e Projetos**, v. 12, n. 1, p. 123–148, 11 mar. 2021. Acesso em: 27 jun. 2025.

TAN, J.; FEITOSA, D.; AVGERIOU, P.; LUNGU, M. Evolution of technical debt remediation in Python: A case study on the Apache Software Ecosystem. **Journal of Software: Evolution and Process**, v. 33, n. 4, 18 nov. 2020.

TAN, J.; FEITOSA, D.; AVGERIOU, P. The lifecycle of Technical Debt that manifests in both source code and issue trackers. **Information and Software Technology**, v. 159, p. 107216–107216, 5 abr. 2023.

VERDECCHIA, R.; KRUCHTEN, P.; LAGO, P.; MALAVOLTA, I. Building and Evaluating a Theory of Architectural Technical Debt in software-intensive Systems. **Journal of Systems and Software**, v. 176, p. 110925, jun. 2021.

VIDONI, M.; CODABUX, Z.; FARD, F. H. Infinite technical debt. **Journal of Systems and Software**, v. 190, p. 111336, ago. 2022.

WIESE, M.; RACHOW, P.; RIEBISCH, M.; SCHWARZE, J. Preventing technical debt with the TAP framework for Technical Debt Aware Management. **Information and Software Technology**, v. 148, p. 106926, ago. 2022.

WIESE, M.; BOROWA, K. IT Managers' Perspective on Technical Debt Management. **Journal of Systems and Software**, p. 111700, abr. 2023.

WIIG, K. Knowledge Management Foundations : Thinking about Thinking : How People and Organizations Create, Represent and Use Knowledge / K.M. Wiig.. Acesso em: 5 maio. 2025.

WOLFART, D.; MARTINEZ, J.; ASSUNÇÃO, W. K. G.; COLANZI, T. E.; EGYED, A. Variability Debt in Opportunistic reuse: a multi-project Field Study. **Journal of Systems and Software**, v. 210, p. 111969, abr. 2024.

YANG, B.; ZHENG, W.; VIERE, C. Holistic Views of Knowledge Management Models. Advances in Developing Human Resources, v. 11, n. 3, p. 273–289, jun. 2009.

ZAZWORKA, N.; VETRO', A.; IZURIETA, C.; WONG, S.; CAI, Y.; SEAMAN, C.; SHULL, F. Comparing Four Approaches for Technical Debt Identification. **Software Quality Journal**, v. 22, n. 3, p. 403–426, 3 abr. 2013.

ZHU, K.; YIN, M.; ZHU, D.; ZHANG, X.; GAO, C.; JIANG, J. SCGRU: a General Approach for Identifying Multiple Classes of self-admitted Technical Debt with Text Generation Oversampling. **Journal of Systems and Software**, v. 195, p. 111514–111514, 17 set. 2022.