ISSN: 2675-7222

SAS
South American Sciences

IMPACTOS DA DÍVIDA TÉCNICA NA MANUTENÇÃO E EVOLUÇÃO DE SOFTWARE

IMPACTS OF TECHNICAL DEBT ON SOFTWARE MAINTENANCE AND EVOLUTION

https://doi.org/10.63330/sasciencesv6n2-339

Submetido em: 17/11/2026 e Publicado em: 26/11/2026

SAS: e26569

Wendril Gabriel Medeiros Holanda
Graduando em Engenharia de Software – 8º período, iCEV.
E-mail: wendril.holanda@somosicev.com
Lattes: https://lattes.cnpq.br/3744771910016839
ORCID: https://orcid.org/0009-0001-5866-063X

Mauro Jose Araujo de Melo
Mestre em Engenharia Elétrica (UFMA)
E-mail: mauro.melo@somosicev.com
Lattes: https://lattes.cnpq.br/5073851179418193

Resumo

A dívida técnica representa compromissos e decisões de desenvolvimento que podem atender necessidades imediatas, mas produzir consequências posteriores para a manutenção e a evolução de sistemas de software. Este trabalho tem como objetivo analisar, com base na literatura acadêmica, os impactos da dívida técnica sobre essas atividades identificados pelos estudos existentes. Trata-se de uma revisão integrativa, de abordagem qualitativa e exploratória, baseada em levantamento estruturado de publicações acadêmicas, com critérios de inclusão e exclusão, triagem por prioridade, avaliação de qualidade, extração de evidências e síntese qualitativa. O corpus reúne 17 estudos, entre revisões e mapeamentos sistemáticos, surveys com profissionais e investigações empíricas. Os resultados indicam impactos recorrentes sobre manutenibilidade, esforço de manutenção e correção, produtividade, retrabalho, custos, degradação estrutural e capacidade de evolução e entrega. Estudos empíricos relacionam maior dívida técnica a maior esforço de manutenção corretiva, perda de tempo de desenvolvimento e custos adicionais, enquanto pesquisas sobre dívida arquitetural destacam dificuldades para implementar novas funcionalidades e realizar mudanças futuras. A literatura também aponta pressão de prazo e foco em entregas rápidas entre fatores associados ao surgimento da dívida. Conclui-se que a dívida técnica pode comprometer a sustentabilidade do software ao aumentar os recursos necessários para manter sistemas existentes e reduzir a eficiência de sua evolução, com impactos inter-relacionados ao longo do ciclo de vida.

Palavras-chave: Dívida técnica; Evolução de software; Manutenção de software.

Abstract

Technical debt represents development compromises and decisions that may address immediate needs but produce later consequences for software maintenance and evolution. This study aims to analyze, based on the academic literature, the impacts of technical debt on software maintenance and evolution identified by existing studies. This is a qualitative and exploratory integrative review based on a structured literature search, with inclusion and exclusion criteria, priority-based screening, quality assessment, evidence extraction, and qualitative synthesis. The corpus comprises 17 studies, including systematic reviews and mappings, practitioner surveys, and empirical investigations. The results indicate recurring impacts on maintainability, maintenance and corrective effort, productivity, rework, costs, structural degradation, and software evolution and delivery capability. Empirical studies relate higher technical debt to greater corrective maintenance effort, loss of development time, and additional costs, while research on architectural technical debt highlights difficulties in implementing new features and performing future changes. The literature also identifies deadline pressure and focus on rapid delivery among factors associated with debt occurrence. The study concludes that technical debt can compromise software sustainability by increasing the resources required to maintain existing systems and reducing the efficiency of software evolution, with interrelated impacts throughout the software life cycle.

Keywords: Software evolution; Software maintenance; Technical debt.

1 INTRODUÇÃO

O desenvolvimento de sistemas de software envolve decisões técnicas tomadas em diferentes momentos do ciclo de vida. Em determinadas situações, as equipes podem adotar soluções que atendem a necessidades imediatas, como o cumprimento de prazos, a redução de esforço ou a viabilização de entregas, mas que podem produzir consequências posteriores. Na Engenharia de Software, esse fenômeno é frequentemente discutido por meio do conceito de dívida técnica, utilizado para representar compromissos técnicos que oferecem benefícios no curto prazo, mas podem tornar mudanças futuras mais custosas e dificultar a manutenção e a evolução dos sistemas (Kruchten; Nord; Ozkaya, 2012; Li; Avgeriou; Liang, 2015).

A literatura indica que as consequências da dívida técnica podem atingir diferentes dimensões do desenvolvimento e da manutenção de software, como atraso nas entregas, baixa manutenibilidade e retrabalho (Ramač *et al.*, 2022), perda de tempo de desenvolvimento (Besker; Martini; Bosch, 2019) e maior esforço de manutenção corretiva (Amanatidis; Chatzigeorgiou; Ampatzoglou, 2017).

Essas evidências mostram que a dívida técnica não deve ser compreendida apenas como um problema interno de código, pois seus efeitos podem alcançar a capacidade de modificar, corrigir e evoluir

sistemas ao longo do tempo. Por isso, investigar seus impactos é relevante para compreender como decisões técnicas presentes podem influenciar custos, esforço, produtividade, manutenibilidade e capacidade de evolução futura do software.

Diante desse contexto, este trabalho investiga os impactos da dívida técnica sobre a manutenção e a evolução de sistemas de software. A pesquisa busca responder à seguinte questão: quais impactos da dívida técnica na manutenção e evolução de sistemas de software são identificados pela literatura? Por se tratar de pesquisa exploratória, não foram formuladas hipóteses prévias. O objetivo geral consiste em analisar esses impactos com base na literatura acadêmica. Para isso, busca-se identificar os principais tipos de dívida técnica abordados na literatura; identificar fatores relacionados ao seu surgimento e acúmulo; analisar seus impactos sobre a manutenção de software; analisar seus efeitos sobre a evolução dos sistemas; e categorizar os principais impactos identificados nos estudos analisados. Tipos, causas e fatores de acúmulo são analisados na medida em que ajudam a compreender em que condições os impactos sobre manutenção e evolução são produzidos ou intensificados nos estudos analisados.

A relevância desta pesquisa está associada à necessidade de compreender de maneira organizada as consequências atribuídas à dívida técnica ao longo do ciclo de vida dos sistemas. A reunião e a análise de evidências provenientes de diferentes estudos podem contribuir para esclarecer de que maneira esse fenômeno interfere nas atividades de manutenção e na capacidade de evolução do software, além de permitir a identificação de padrões recorrentes em diferentes contextos de pesquisa.

A literatura sobre dívida técnica apresenta ampla diversidade de tipos, estratégias de gerenciamento e contextos de investigação, conforme evidenciado por mapeamentos e revisões anteriores (Li; Avgeriou; Liang, 2015; Lenarduzzi *et al.*, 2021; Melo *et al.*, 2022). Nesse cenário, a lacuna abordada por este trabalho não consiste na ausência de estudos sobre dívida técnica, mas na necessidade de reunir e organizar, sob o recorte específico de manutenção e evolução de software, impactos que aparecem distribuídos em estudos com diferentes objetos, métodos e formas de mensuração. A contribuição pretendida é, portanto, uma síntese analítica orientada pela pergunta de pesquisa e pelos objetivos definidos.

Este artigo está organizado da seguinte forma: após esta introdução, o desenvolvimento apresenta o conceito de dívida técnica, seus principais tipos, fatores relacionados ao surgimento e ao acúmulo e sua relação com manutenção e evolução de software. Em seguida, é apresentada a metodologia utilizada para o levantamento e a análise da literatura. Posteriormente, são apresentados e discutidos os tipos e as causas de dívida técnica relacionados aos impactos e os impactos identificados nos estudos selecionados, organizados em categorias e em um modelo interpretativo. Por fim, as considerações finais retomam a pergunta de pesquisa e os objetivos do trabalho à luz das evidências analisadas.

2 REFERENCIAL TEÓRICO

2.1 DÍVIDA TÉCNICA

O conceito de dívida técnica é utilizado na Engenharia de Software para representar situações em que decisões ou soluções técnicas favorecem necessidades imediatas, mas podem gerar consequências futuras para o sistema. A metáfora foi proposta por Cunningham (1992), que comparou a entrega de código ainda imaturo à contração de uma dívida: o atalho acelera a entrega, mas gera juros enquanto o código não é reescrito. Kruchten, Nord e Ozkaya (2012) discutem a dívida técnica como um conceito que ultrapassa a simples metáfora financeira e está relacionado a decisões de desenvolvimento capazes de influenciar o custo e a dificuldade de mudanças posteriores.

Uma característica central desse conceito é a relação entre benefício de curto prazo e consequências de longo prazo. Li, Avgeriou e Liang (2015), em um mapeamento sistemático da literatura, descrevem a dívida técnica como resultado de compromissos técnicos que podem oferecer vantagens imediatas, mas prejudicar a saúde do software no futuro. O estudo também evidencia que o conceito não está restrito ao código, abrangendo diferentes tipos de dívida relacionados a artefatos e atividades do desenvolvimento.

Alves *et al.* (2016) apresentam perspectiva semelhante ao relacionar a dívida técnica a artefatos de software que podem gerar benefícios imediatos, mas produzir consequências posteriores, especialmente durante atividades de manutenção. Ao analisar a literatura sobre identificação e gerenciamento de dívida técnica, os autores mostram que o fenômeno pode assumir diferentes formas e demandar estratégias específicas de identificação e tratamento.

Tom, Aurum e Vidgen (2013) também destacam a amplitude do conceito ao investigar suas diferentes dimensões, antecedentes e consequências. O estudo reforça que a dívida técnica não corresponde simplesmente a qualquer problema existente no software, sendo necessário considerar as decisões, os compromissos técnicos envolvidos e seus possíveis efeitos futuros.

Essas contribuições convergem para a definição consensual estabelecida no Dagstuhl Seminar 16162, segundo a qual, em sistemas intensivos em software, a dívida técnica é um conjunto de construções de design ou de implementação convenientes no curto prazo, mas que criam um contexto técnico capaz de tornar mudanças futuras mais custosas ou impossíveis. Trata-se de um passivo real ou contingente, cujo impacto se limita às qualidades internas do sistema, principalmente à manutenibilidade e à evolutibilidade (Avgeriou *et al.*, 2016). Este trabalho adota essa definição, que vincula explicitamente a dívida técnica à manutenção e à evolução de software.

2.2 TIPOS DE DÍVIDA TÉCNICA

A literatura mostra que a dívida técnica pode assumir diferentes formas ao longo do ciclo de desenvolvimento. No mapeamento sistemático de Li, Avgeriou e Liang (2015), realizado a partir de 94

estudos publicados entre 1992 e 2013, a dívida técnica foi organizada em dez tipos: dívida de requisitos, dívida arquitetural, dívida de design, dívida de código, dívida de testes, dívida de build, dívida de documentação, dívida de infraestrutura, dívida de versionamento e dívida de defeitos. Entre essas categorias, a dívida de código recebeu maior atenção nos estudos analisados, enquanto outras categorias apresentavam menor volume de investigação.

A classificação proposta por Li, Avgeriou e Liang (2015) evidencia que a dívida técnica não se limita à implementação. A dívida de requisitos relaciona-se às diferenças entre uma especificação considerada adequada e aquilo que é efetivamente implementado; a dívida arquitetural decorre de decisões de arquitetura que comprometem atributos internos de qualidade; a dívida de design envolve atalhos ou compromissos no desenho detalhado; e a dívida de código inclui práticas como duplicação ou complexidade excessiva. Os autores também identificam dívidas associadas a testes insuficientes, processos de build inadequados, documentação desatualizada ou incompleta, infraestrutura subótima, problemas de versionamento e defeitos tratados na literatura como itens de dívida.

Alves *et al.* (2016), em outro mapeamento sistemático, analisaram 100 estudos e propuseram uma taxonomia própria de tipos de dívida técnica, além de indicadores e estratégias para identificação e gerenciamento. A existência de diferentes classificações reforça que o conceito é amplo e pode ser aplicado a diferentes artefatos e atividades do desenvolvimento. Para este trabalho, os tipos de dívida são relevantes porque permitem observar se determinados impactos sobre manutenção e evolução aparecem associados com maior frequência a categorias específicas, como dívida de código, arquitetura, testes ou documentação.

Em um estudo terciário, Rios, Mendonça Neto e Spínola (2018) analisaram revisões e mapeamentos secundários sobre dívida técnica e consolidaram uma taxonomia de tipos, um panorama de estratégias de gerenciamento e tendências de pesquisa na área. Ao reunir resultados de diferentes estudos secundários, o trabalho oferece uma base mais ampla do que classificações isoladas e confirma que a dívida técnica se distribui por diversos artefatos do ciclo de desenvolvimento.

A literatura específica sobre dívida de requisitos também reforça a necessidade de não reduzir o fenômeno ao código. Melo *et al.* (2022), em uma revisão sistemática que partiu de 7.499 registros e incluiu 66 estudos primários, observaram que a dívida técnica de requisitos ainda constitui um campo relativamente incipiente, especialmente quanto à identificação e mensuração. Os autores relacionam esse tipo de dívida à distância entre uma especificação considerada adequada e a implementação efetivamente realizada e destacam que sua gestão é relevante para a evolução do software. Essa evidência amplia o recorte dos tipos de dívida e mostra que problemas em artefatos anteriores à implementação podem influenciar mudanças posteriores.

2.3 SURGIMENTO E ACÚMULO DE DÍVIDA TÉCNICA

O surgimento da dívida técnica pode estar associado a diferentes condições técnicas e organizacionais. Em uma revisão sistemática sobre dívida técnica no desenvolvimento ágil, Behutiye *et al.* (2017) identificaram oito categorias de causas. Entre as mais recorrentes estavam o foco em entregas rápidas e problemas de arquitetura e design. Esse resultado indica que a pressão por resultados de curto prazo pode favorecer decisões que transferem parte do esforço técnico para momentos posteriores do desenvolvimento.

Evidências provenientes da indústria reforçam essa relação. O projeto InsighTD, família internacional de surveys sobre causas e efeitos da dívida técnica iniciada no Brasil, identificou prazos e pressão por entregas entre as causas mais citadas por profissionais desde seus primeiros resultados (Rios *et al.*, 2018, 2020). Esses estudos mostram também que a dívida técnica pode surgir em diferentes atividades do desenvolvimento, e não apenas na codificação, o que amplia a necessidade de observar requisitos, arquitetura, testes, documentação e outros artefatos durante a análise do fenômeno.

O acúmulo da dívida técnica ocorre quando itens existentes permanecem sem tratamento e novos itens são introduzidos ao longo da evolução do software. Esse comportamento pode ser observado durante atividades rotineiras de desenvolvimento (Calixto; Araújo; Alves, 2024), como se discute na seção 4.2.

A literatura também indica que o gerenciamento da dívida técnica nem sempre ocorre de maneira sistemática. Martini, Besker e Bosch (2018) observaram que o rastreamento da dívida técnica raramente é feito de forma sistemática, aspecto detalhado na seção 4.1. Esse cenário ajuda a compreender por que itens de dívida podem permanecer no sistema e se tornar parte do trabalho futuro de manutenção e evolução.

2.4 MANUTENÇÃO E EVOLUÇÃO DE SOFTWARE

A manutenção de software compreende o conjunto de atividades necessárias para sustentar um sistema ao longo de seu ciclo de vida. A norma ISO/IEC/IEEE 14764:2022 (International Organization for Standardization, 2022) estabelece diretrizes para o processo de manutenção, incluindo suas atividades, tarefas e diferentes tipos. Nesse contexto, a manutenção não se restringe à correção de defeitos após a entrega, mas envolve ações necessárias para conservar ou adaptar o software diante de mudanças e novas necessidades.

Bennett e Rajlich (2000) discutem manutenção e evolução como atividades intimamente relacionadas ao desenvolvimento de sistemas existentes. Para os autores, sistemas de software continuam sendo modificados após sua entrega em razão de problemas identificados, necessidades de melhoria e mudanças no ambiente ou nos requisitos. A evolução, nesse sentido, está associada à capacidade de um sistema continuar sendo modificado de maneira viável enquanto permanece útil.

Lehman (1980) formulou leis de evolução de software segundo as quais sistemas em uso precisam

ser continuamente modificados para permanecerem úteis (lei da mudança contínua) e, à medida que evoluem, tendem a ter sua complexidade aumentada, a menos que se realize trabalho específico para mantê-la ou reduzi-la (lei da complexidade crescente). A manutenibilidade, por sua vez, é definida pela norma ISO/IEC 25010 como o grau de eficácia e eficiência com que um produto pode ser modificado por seus mantenedores, e compreende as subcaracterísticas modularidade, reusabilidade, analisabilidade, modificabilidade e testabilidade (International Organization for Standardization, 2023). Neste trabalho, o termo manutenibilidade é utilizado nesse sentido.

Essa relação é especialmente importante para o estudo da dívida técnica. Decisões que degradam a estrutura interna, reduzem a qualidade de artefatos ou aumentam dependências podem não impedir imediatamente o funcionamento do software, mas podem tornar modificações posteriores mais difíceis ou custosas. Assim, manutenção e evolução constituem dimensões adequadas para observar os efeitos de longo prazo da dívida técnica, pois concentram atividades de correção, adaptação, melhoria e introdução de novas capacidades no sistema.

2.5 RELAÇÕES CONCEITUAIS ENTRE DÍVIDA TÉCNICA, MANUTENÇÃO E EVOLUÇÃO

A metáfora financeira oferece uma chave conceitual para compreender como a dívida técnica afeta a manutenção e a evolução. O principal corresponde ao esforço necessário para remover um item de dívida, enquanto os juros correspondem ao esforço adicional exigido pelas mudanças realizadas enquanto o item permanece no sistema (Avgeriou *et al.*, 2016; Kruchten; Nord; Ozkaya, 2012). Sob essa perspectiva, os impactos da dívida manifestam-se principalmente como juros pagos nas atividades de manutenção e evolução.

Como a definição adotada limita o impacto da dívida às qualidades internas do sistema (Avgeriou *et al.*, 2016), seus efeitos tendem a aparecer de forma mediada. A degradação de atributos como modularidade, analisabilidade e modificabilidade (International Organization for Standardization, 2023) aumenta o esforço necessário para compreender, corrigir e modificar o software, e esse esforço adicional pode se converter em perda de produtividade, retrabalho e custos.

Do ponto de vista da evolução, a dívida técnica pode ser compreendida como um fator que acentua a tendência de aumento da complexidade descrita por Lehman (1980). Quando as mudanças contínuas exigidas pelo uso do sistema são realizadas sem trabalho de preservação da estrutura, os juros acumulados podem reduzir a capacidade de introduzir novas funcionalidades e de realizar entregas no ritmo esperado.

Essas relações conceituais orientam a análise, mas não são tratadas como hipóteses causais a serem testadas. As categorias de impacto utilizadas neste trabalho derivam da síntese do corpus e são apresentadas no capítulo 4.

3 METODOLOGIA

Este trabalho caracteriza-se como uma pesquisa bibliográfica do tipo revisão integrativa da literatura, de abordagem qualitativa e exploratória. O levantamento e a seleção de trabalhos acadêmicos foram orientados para estudos sobre dívida técnica, manutenção e evolução de software. Os estudos selecionados foram analisados para identificar tipos de dívida técnica, fatores associados ao seu surgimento e acúmulo e, principalmente, impactos relatados sobre a manutenção e a evolução dos sistemas.

A revisão foi conduzida por meio de um levantamento estruturado da literatura, orientado pela pergunta de pesquisa e pelos objetivos definidos. Como referência metodológica para a condução das etapas, sem a pretensão de exaustividade de uma revisão sistemática, foram consideradas as recomendações de Kitchenham e Charters (2007) para revisões de literatura em Engenharia de Software, especialmente quanto à definição da questão de pesquisa, estratégia de busca, critérios de seleção, extração de dados e síntese das evidências. O snowballing, conforme as diretrizes de Wohlin (2014), foi utilizado de forma complementar para localizar estudos relacionados a partir de referências, citações e trabalhos centrais já identificados.

As buscas foram realizadas em agosto de 2026 nas bases ScienceDirect/Elsevier, ACM Digital Library, IEEE Xplore, Software Engineering Institute e Sociedade Brasileira de Computação, com apoio do índice bibliográfico DBLP. Os dados bibliográficos de cada estudo foram confirmados pelo DOI ou pela página oficial do veículo de publicação.

Ao final de agosto de 2026, uma busca complementar voltada a estudos recentes e a grupos de autoria ainda não representados no corpus resultou na inclusão de Wolfart *et al.* (2024).

A estratégia de busca foi organizada em dois blocos de termos. O primeiro representou o fenômeno investigado, incluindo expressões como "technical debt", "architectural technical debt", "code debt", "design debt", "test debt", "documentation debt" e "requirements technical debt". O segundo bloco representou os efeitos de interesse, utilizando termos como maintenance, maintainability, software evolution, evolvability, modifiability, productivity, rework, maintenance cost, complexity, defect e delivery delay. A string genérica, adaptada à sintaxe de cada mecanismo de busca, foi: ("technical debt" OR "architectural technical debt" OR "code debt" OR "design debt" OR "test debt" OR "documentation debt" OR "requirements technical debt") AND (maintenance OR maintainability OR "software evolution" OR evolvability OR modifiability OR productivity OR rework OR "maintenance cost" OR complexity OR defect OR "delivery delay").

Foram considerados elegíveis estudos acadêmicos que: a) tratassem explicitamente de dívida técnica; b) apresentassem contribuição para compreender manutenção, evolução, manutenibilidade, produtividade, custo, retrabalho, defeitos, complexidade, arquitetura ou capacidade de mudança; c) contribuíssem para identificar tipos, causas ou fatores de acúmulo quando relacionados aos objetivos

específicos; d) apresentassem metadados bibliográficos verificáveis; e e) disponibilizassem informação suficiente para avaliar objetivo, método e resultados. Trabalhos fundamentais de natureza conceitual também foram mantidos quando necessários para a definição do fenômeno.

Foram excluídos resultados duplicados, publicações sem relação efetiva com a pergunta de pesquisa, materiais não acadêmicos utilizados apenas como opinião ou divulgação, estudos centrados exclusivamente em ferramentas ou técnicas de detecção sem evidência útil para os objetivos desta pesquisa e registros cuja autoria, veículo ou conteúdo não puderam ser confirmados.

A seleção foi realizada em duas etapas. Primeiro, título, resumo e metadados foram avaliados quanto à aderência ao tema e à pergunta de pesquisa. Cada estudo recebeu, nessa etapa, uma classificação de prioridade: alta, quando atendia aos critérios a, b, d e e e apresentava evidência direta, empírica ou de síntese secundária, sobre impactos na manutenção ou na evolução; média, quando atendia aos critérios a, d e e e contribuía para tipos, causas ou fatores de acúmulo (critério c) ou apresentava evidência apenas indireta sobre impactos; e baixa, nos demais casos, que foram excluídos. Em seguida, os estudos de alta e média prioridade foram lidos na íntegra, observando-se o tipo de pesquisa, o contexto ou a amostra, as evidências apresentadas, os resultados, as limitações e a contribuição para os objetivos desta pesquisa.

Para cada estudo selecionado, foram extraídas, quando disponíveis, as seguintes informações: autoria, ano, título, veículo, DOI, objetivo, método, amostra ou contexto, principais resultados, tipos ou causas de dívida técnica, impactos relacionados à manutenção e à evolução e limitações. A síntese foi realizada qualitativamente, comparando convergências e divergências entre os estudos. Os impactos foram agrupados por similaridade semântica em categorias analíticas, mantendo-se separadas as afirmações originalmente apresentadas pelas fontes e as interpretações produzidas nesta pesquisa.

A avaliação de qualidade dos estudos selecionados considerou três critérios: QA1, publicação em periódico ou evento com revisão por pares e DOI verificável; QA2, descrição explícita do objetivo, do método e do contexto ou da amostra; e QA3, natureza da evidência, classificada como síntese secundária (S), percepção de profissionais (P) ou métricas e dados de projetos (D). Todos os estudos do corpus atenderam a QA1 e QA2. O critério QA3 não foi utilizado para exclusão, mas para ponderar a força das evidências na síntese, conforme os Quadros 1 e 2.

O corpus final é composto por 17 estudos, caracterizados no Quadro 1, aos quais se somam obras conceituais e normativas utilizadas no referencial teórico. Como o número de resultados por base não foi registrado de modo uniforme nas buscas iniciais, não é apresentado fluxograma no modelo PRISMA 2020 com contagens por etapa, o que é compatível com a natureza integrativa da revisão.

4 RESULTADOS E DISCUSSÃO

4.1 CARACTERIZAÇÃO DOS ESTUDOS ANALISADOS

O corpus analítico reuniu 17 estudos: seis sínteses secundárias (revisões e mapeamentos sistemáticos), seis estudos baseados em métricas ou dados de projetos, quatro estudos baseados na percepção de profissionais e um estudo de campo que combina percepção e dados de artefatos, conforme o Quadro 1. Essa diversidade metodológica foi considerada importante porque a pergunta de pesquisa envolve impactos que podem ser observados tanto por métricas e dados de manutenção quanto pela percepção de profissionais envolvidos no desenvolvimento.

Entre as sínteses secundárias, os mapeamentos de Li, Avgeriou e Liang (2015) e de Alves *et al.* (2016), apresentados na seção 2.2, fornecem a base para a identificação dos tipos de dívida. Behutiye *et al.* (2017), em uma revisão sistemática com 38 estudos primários no contexto ágil, identificaram causas e consequências recorrentes, entre elas redução de produtividade, degradação do sistema e aumento do custo de manutenção.

Entre os estudos empíricos, Amanatidis, Chatzigeorgiou e Ampatzoglou (2017) analisaram dez projetos PHP de código aberto para investigar a relação entre quantidade de dívida técnica e manutenção corretiva. Os resultados mostraram relação positiva entre o volume de dívida e a frequência e o esforço de correção, indicando que módulos com maior dívida tendem a ser mais propensos a defeitos e mais custosos de manter. Conejero *et al.* (2018) analisaram três linhas de produto de software e encontraram correlações entre anomalias de modularidade e atributos de manutenibilidade, especialmente estabilidade.

Mendes *et al.* (2016) investigaram dívida de documentação de requisitos em um projeto real que utilizava requisitos ágeis. O estudo analisou 132 tarefas de manutenção e evolução, das quais 65 estavam relacionadas à presença de dívida de documentação em um período de 18 meses. Os autores relataram esforço adicional de manutenção equivalente a aproximadamente 47% do esforço inicialmente estimado para o desenvolvimento do projeto e custo adicional de cerca de 48% do custo inicial da fase de desenvolvimento.

Martini, Besker e Bosch (2018) combinaram um survey com 226 participantes de 15 organizações e estudos de caso em três organizações para investigar o rastreamento e o custo do gerenciamento da dívida técnica. O estudo estimou que, em média, 25% do tempo de desenvolvimento era dedicado ao gerenciamento da dívida técnica; ao mesmo tempo, apenas 26% dos participantes utilizavam alguma ferramenta para rastreá-la e 7,2% declararam fazê-lo de forma metódica. Besker, Martini e Bosch (2019), em estudo longitudinal com 43 desenvolvedores, 16 entrevistas e uma etapa de replicação com conjunto de dados independente, verificaram perda média relatada de 23% do tempo de trabalho devido a problemas relacionados à dívida técnica, com atividades adicionais de teste, análise de código e refatoração.

No contexto arquitetural, Besker, Martini e Bosch (2018) sintetizaram a literatura sobre dívida técnica arquitetural e destacaram seus efeitos sobre manutenção e evolução. Verdecchia *et al.* (2021), por meio de grounded theory seguida de grupos focais, construíram e avaliaram uma teoria empiricamente fundamentada sobre dívida arquitetural, descrevendo causas, sintomas, consequências e estratégias de gerenciamento em organizações de software. A dívida arquitetural é caracterizada nesse trabalho por decisões de grande escala que podem ser adequadas quando adotadas, mas posteriormente dificultar significativamente o progresso do sistema.

Ramač *et al.* (2022) ampliaram a evidência industrial por meio de uma família internacional de surveys, reunindo 653 respostas válidas de seis países. A pressão de tempo ou prazo foi a causa individual mais citada, enquanto atraso de entrega, baixa manutenibilidade e retrabalho foram os efeitos mais frequentes. Em estudo brasileiro mais recente, Calixto, Araújo e Alves (2024) analisaram 2.035 pull requests de 12 projetos Java da Apache e observaram que itens de dívida técnica podem ser reduzidos, mantidos ou aumentados durante atividades rotineiras de evolução do software.

Outras evidências empíricas reforçam a relação entre estrutura e custo de manutenção. MacCormack e Sturtevant (2016) analisaram dois sistemas de software de porte semelhante e estruturas arquiteturais distintas, abrangendo mais de 20 mil componentes, e verificaram que componentes fortemente acoplados apresentavam custos de manutenção, medidos pela atividade relacionada a defeitos, significativamente maiores do que componentes periféricos menos acoplados. O resultado fornece evidência de que características arquiteturais associadas à dívida podem ser observadas em atividade de manutenção efetiva, e não apenas em percepção de profissionais.

Paudel, Gonzalez-Huerta e Zabardast (2026) analisaram dados de dívida técnica de um produto industrial composto por um monólito e 78 microsserviços, no período de agosto de 2022 a dezembro de 2024. O estudo não estabelece causalidade entre estilo arquitetural e dívida técnica, mas mostra que a densidade de dívida evoluiu de forma distinta entre o monólito e os microsserviços analisados, reforçando a necessidade de interpretação contextual e acompanhamento contínuo durante a evolução.

Wolfart *et al.* (2024) conduziram um estudo de campo em uma empresa multinacional, envolvendo três sistemas, sete variantes e 22 participantes. No contexto de reutilização oportunista, os autores observaram duplicação de código, artefatos não sincronizados e dificuldades para a evolução e a manutenção de variantes, além de identificarem a pressão de tempo como causa relevante para a adoção dessas soluções. Por se tratar de um contexto específico de dívida de variabilidade, seus resultados não são generalizados para outros tipos de dívida.

Quadro 1 – Caracterização dos estudos que compõem o corpus

Estudo	Método/contexto	Tipo	Evidência principal
Li, Avgeriou e Liang (2015)	Mapeamento sistemático; 94 estudos	S	Dez tipos de dívida técnica e panorama de gerenciamento
Alves *et al.* (2016)	Mapeamento sistemático; 100 estudos	S	Taxonomia, indicadores e estratégias de gerenciamento
Behutiye *et al.* (2017)	Revisão sistemática; 38 estudos, contexto ágil	S	Redução de produtividade, degradação do sistema e aumento do custo de manutenção
Besker, Martini e Bosch (2018)	Revisão sistemática; dívida arquitetural	S	Pode dificultar novas funcionalidades e elevar o custo da manutenção arquitetural
Lenarduzzi *et al.* (2021)	Revisão sistemática; 44 estudos primários	S	Falta de evidência consolidada para mensurar principal e juros e priorizar a dívida
Melo *et al.* (2022)	Revisão sistemática; 66 estudos primários	S	Dívida de requisitos pouco explorada e relevante para a evolução
Mendes *et al.* (2016)	Estudo retrospectivo; 132 tarefas, 18 meses	D	Esforço e custo adicionais associados à dívida de documentação de requisitos
MacCormack e Sturtevant (2016)	Dois sistemas; mais de 20 mil componentes	D	Alto acoplamento associado a maior custo de manutenção
Amanatidis, Chatzigeorgiou e Ampatzoglou (2017)	Estudo empírico; 10 projetos PHP	D	Maior dívida associada a maior frequência e esforço de manutenção corretiva
Conejero *et al.* (2018)	Estudo empírico; três linhas de produto	D	Anomalias de modularidade relacionadas a atributos de manutenibilidade
Calixto, Araújo e Alves (2024)	Estudo empírico; 2.035 pull requests de 12 projetos	D	Itens de dívida podem diminuir, permanecer ou aumentar durante mudanças
Paudel, Gonzalez-Huerta e Zabardast (2026)	Caso industrial; monólito e 78 microsserviços	D	Densidade de dívida evolui de forma distinta conforme o contexto arquitetural
Martini, Besker e Bosch (2018)	Survey e casos múltiplos; 226 participantes	P	Gestão da dívida consome parcela relevante do tempo; rastreamento pouco sistemático
Besker, Martini e Bosch (2019)	Estudo longitudinal e replicação; 43 desenvolvedores	P	Perda média relatada de 23% do tempo de trabalho e atividades adicionais
Verdecchia *et al.* (2021)	Grounded theory e grupos focais	P	Decisões arquiteturais podem dificultar o progresso futuro do sistema
Ramač *et al.* (2022)	Família de surveys; 653 respostas, seis países	P	Atraso de entrega, baixa manutenibilidade e retrabalho entre os efeitos mais citados
Wolfart *et al.* (2024)	Estudo de campo; 3 sistemas, 7 variantes, 22 participantes	P e D	Duplicação e artefatos não sincronizados dificultam a manutenção e a evolução de variantes

Fonte: elaborado pelos autores (2026), com base nos estudos analisados. Nota: S = síntese secundária; P = percepção de profissionais; D = métricas ou dados de projetos.

4.2 TIPOS E CAUSAS DA DÍVIDA TÉCNICA ASSOCIADOS AOS IMPACTOS

Quanto aos tipos, o corpus concentra-se em dívida de código e dívida arquitetural, o que é coerente com os mapeamentos de Li, Avgeriou e Liang (2015) e com o estudo terciário de Rios, Mendonça Neto e Spínola (2018). A relação entre tipo e impacto, no entanto, não é uniforme. A dívida arquitetural aparece associada principalmente à capacidade de evolução, à degradação estrutural e ao custo de manutenção (Besker; Martini; Bosch, 2018; MacCormack; Sturtevant, 2016; Paudel; Gonzalez-Huerta; Zabardast, 2026; Verdecchia *et al.*, 2021). A dívida de código e de design relaciona-se sobretudo à manutenibilidade e ao esforço de manutenção corretiva (Amanatidis; Chatzigeorgiou; Ampatzoglou, 2017; Conejero *et al.*, 2018). A dívida de documentação e de requisitos aparece associada a esforço e custo adicionais e a dificuldades de evolução (Melo *et al.*, 2022; Mendes *et al.*, 2016), enquanto a dívida de variabilidade se relaciona à duplicação e à dificuldade de manter variantes sincronizadas (Wolfart *et al.*, 2024).

Quanto às causas, a pressão de prazo e o foco em entregas rápidas são os fatores mais recorrentes, identificados tanto em síntese da literatura (Behutiye *et al.*, 2017) quanto em evidências industriais do corpus (Ramač *et al.*, 2022; Wolfart *et al.*, 2024), em consonância com os primeiros resultados do projeto InsighTD (Rios *et al.*, 2020). Essas causas conectam-se diretamente à categoria de capacidade de evolução e entrega: a dívida contraída para acelerar uma entrega tende a gerar juros que posteriormente atrasam outras entregas, efeito que figura entre os mais citados por profissionais (Ramač *et al.*, 2022). Problemas de arquitetura e design, também apontados como causa por Behutiye *et al.* (2017), relacionam-se à degradação estrutural e à manutenibilidade.

Quanto ao acúmulo, três mecanismos aparecem no corpus: o rastreamento pouco sistemático, que permite a permanência de itens sem tratamento (Martini; Besker; Bosch, 2018); a modificação da dívida durante mudanças rotineiras (Calixto; Araújo; Alves, 2024); e a introdução de nova dívida em razão de dívida já existente (Besker; Martini; Bosch, 2019). Esses mecanismos ajudam a explicar por que os impactos sobre manutenção e evolução tendem a se prolongar no tempo e fundamentam a relação de retroalimentação representada na Figura 1.

4.3 IMPACTOS SOBRE MANUTENIBILIDADE E ESFORÇO DE MANUTENÇÃO

A manutenibilidade aparece entre os impactos identificados tanto pela percepção de profissionais (Ramač *et al.*, 2022) quanto por métricas de modularidade (Conejero *et al.*, 2018). Amanatidis, Chatzigeorgiou e Ampatzoglou (2017) acrescentam uma dimensão operacional, ao associar maior volume de dívida a maior esforço de manutenção corretiva.

A evidência arquitetural de MacCormack e Sturtevant (2016) complementa esse conjunto. A convergência entre survey, métricas de modularidade, dados de manutenção corretiva e análise arquitetural fortalece essas duas categorias: o esforço de manutenção reúne seis estudos, três deles exclusivamente

baseados em dados de projetos, e a manutenibilidade, embora citada diretamente por apenas dois estudos, combina evidências de naturezas distintas (Quadro 2).

A convergência entre essas evidências sugere que a dívida técnica pode afetar a manutenção por diferentes mecanismos. Em alguns estudos, o problema aparece como deterioração de atributos internos que tornam o sistema mais difícil de modificar; em outros, é observado diretamente como maior frequência de correções e maior esforço para tratar defeitos. A síntese desta pesquisa, portanto, separa a manutenibilidade, entendida como atributo de qualidade, do esforço de manutenção, tratado como consequência operacional, embora as duas categorias estejam relacionadas.

4.4 IMPACTOS SOBRE PRODUTIVIDADE, RETRABALHO E CUSTO

A produtividade constitui outra categoria com sustentação relevante no corpus analisado. A redução de produtividade figura entre as principais consequências sintetizadas por Behutiye *et al.* (2017). A perda de tempo relatada por desenvolvedores (Besker; Martini; Bosch, 2019) oferece a evidência mais direta dessa categoria. Martini, Besker e Bosch (2018) complementam esse quadro ao estimar a parcela do tempo de desenvolvimento dedicada à gestão da dívida.

O retrabalho figura entre os efeitos mais citados por profissionais (Ramač *et al.*, 2022), enquanto o caso de dívida de documentação analisado por Mendes *et al.* (2016) mostra que o esforço e o custo adicionais associados à dívida podem assumir dimensão econômica relevante. Esses resultados sustentam a interpretação de que produtividade, retrabalho e custo representam dimensões distintas de um mesmo problema: recursos que poderiam ser direcionados à evolução do produto passam a ser consumidos para compreender, contornar ou corrigir consequências de decisões anteriores.

4.5 IMPACTOS SOBRE EVOLUÇÃO E CAPACIDADE DE MUDANÇA

Os efeitos sobre a evolução são especialmente visíveis em estudos de dívida arquitetural. Besker, Martini e Bosch (2018) destacam que esse tipo de dívida pode dificultar a implementação de novas funcionalidades e gerar esforços de manutenção arquitetural elevados. A teoria de Verdecchia *et al.* (2021) reforça essa relação e acrescenta que a dívida arquitetural pode exigir remediações de grande alcance.

A literatura sobre priorização também evidencia que as decisões de tratamento da dívida competem com o desenvolvimento de funcionalidades e a correção de defeitos. Lenarduzzi *et al.* (2021), em revisão sistemática de 44 estudos primários selecionados a partir de 557 trabalhos únicos, observaram forte concentração de pesquisas em dívida de código e arquitetural e destacaram falta de evidência empírica consolidada para mensuração de principal e juros da dívida. Esse resultado impõe cautela: embora os impactos da dívida sejam recorrentes, a literatura ainda apresenta limitações para quantificar de modo uniforme sua magnitude e para determinar quando uma dívida específica deve ser priorizada.

A evolução também se relaciona ao comportamento da dívida durante mudanças rotineiras. Como discutido na seção 4.2, as próprias atividades de evolução modificam a dívida. Assim, a evolução do software não é apenas afetada pela dívida já existente; ela também constitui um momento em que a própria dívida é modificada. No caso da reutilização oportunista, a duplicação e as inconsistências entre artefatos tornam a sincronização e a evolução de variantes mais complexas (Wolfart *et al.*, 2024).

4.6 SÍNTESE DAS CATEGORIAS DE IMPACTO

A comparação dos estudos permitiu organizar os impactos identificados em sete categorias principais: manutenibilidade; esforço de manutenção e correção; produtividade; retrabalho; custos; degradação estrutural; e capacidade de evolução e entrega. Essas categorias não foram tomadas de uma única taxonomia existente. Elas constituem uma síntese produzida nesta pesquisa a partir da recorrência e proximidade semântica dos efeitos reportados nas fontes analisadas. O Quadro 2 relaciona cada estudo às categorias, indicando o tipo de evidência e a frequência de cada categoria no corpus.

A força das evidências não é uniforme entre as categorias. Degradação estrutural (oito estudos) e capacidade de evolução e entrega (sete) são as categorias mais frequentes, seguidas de esforço de manutenção e correção (seis) e custos (cinco); produtividade (três), manutenibilidade (dois) e retrabalho (dois) aparecem em menos estudos. A frequência, contudo, não equivale à força da evidência. O esforço de manutenção é sustentado por três estudos exclusivamente baseados em dados de projetos, e a manutenibilidade, embora menos frequente, combina percepção de profissionais e métricas estruturais. A produtividade é sustentada por evidências obtidas diretamente com desenvolvedores e por síntese da literatura. Os custos apresentam resultados quantitativos em contextos específicos, como dívida de documentação e acoplamento arquitetural, que não devem ser generalizados como percentuais universais. O retrabalho aparece apenas em evidências baseadas na percepção de profissionais, enquanto degradação estrutural e capacidade de evolução são sustentadas sobretudo por estudos de arquitetura, modularidade e evolução da dívida. Essa distinção é importante para evitar que todas as categorias sejam tratadas como se apresentassem o mesmo nível de sustentação empírica.

A categoria de manutenibilidade reúne evidências relacionadas à estabilidade, à facilidade de modificação e à dificuldade de manutenção. O esforço de manutenção concentra resultados sobre frequência de correções, necessidade de testes, análise adicional e outras atividades exigidas para lidar com a dívida. A produtividade representa a perda de tempo ou de capacidade de desenvolvimento, enquanto o retrabalho corresponde à repetição ou à realização de atividades adicionais. Os custos abrangem recursos econômicos e esforço convertido em custo. A degradação estrutural inclui efeitos sobre modularidade, arquitetura e qualidade interna. Por fim, a capacidade de evolução e entrega reúne dificuldades relacionadas à introdução de funcionalidades, à realização de mudanças futuras e aos atrasos na entrega.

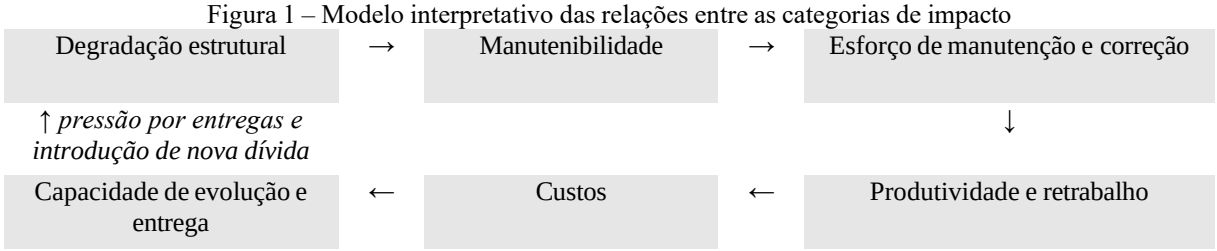
Quadro 2 – Matriz de relação entre os estudos e as categorias de impacto

Estudo	Tipo	MAN	ESF	PRO	RET	CUS	DEG	EVO
Behutiye *et al.* (2017)	S			X		X	X	
Besker, Martini e Bosch (2018)	S		X			X	X	X
Melo *et al.* (2022)	S							X
Mendes *et al.* (2016)	D		X			X		
MacCormack e Sturtevant (2016)	D		X			X	X	
Amanatidis, Chatzigeorgiou e Ampatzoglou (2017)	D		X					
Conejero *et al.* (2018)	D	X					X	
Calixto, Araújo e Alves (2024)	D						X	X
Paudel, Gonzalez-Huerta e Zabardast (2026)	D						X	X
Martini, Besker e Bosch (2018)	P			X		X		
Besker, Martini e Bosch (2019)	P		X	X	X			
Verdecchia *et al.* (2021)	P						X	X
Ramač *et al.* (2022)	P	X			X			X
Wolfart *et al.* (2024)	P e D		X				X	X
Frequência (n = 14)	—	**2**	**6**	**3**	**2**	**5**	**8**	**7**

Fonte: elaborado pelos autores (2026), com base nos estudos analisados. Nota: MAN = manutenibilidade; ESF = esforço de manutenção e correção; PRO = produtividade; RET = retrabalho; CUS = custos; DEG = degradação estrutural; EVO = capacidade de evolução e entrega. Tipo de evidência: S = síntese secundária (revisão ou mapeamento); P = percepção de profissionais; D = métricas ou dados de projetos. Li, Avgeriou e Liang (2015), Alves *et al.* (2016) e Lenarduzzi *et al.* (2021) não constam da matriz por contribuírem para tipos, gerenciamento e priorização, e não para categorias específicas de impacto.

A síntese dos resultados indica que essas categorias podem se relacionar, mas não permite estabelecer uma sequência causal única entre elas. Em diferentes estudos, degradação estrutural, menor manutenibilidade, aumento do esforço de correção, perda de produtividade, elevação de custos e dificuldades de evolução aparecem associados por mecanismos e contextos distintos. A aproximação entre essas categorias constitui uma síntese interpretativa desta pesquisa, construída a partir da convergência parcial entre as evidências, e não um modelo causal validado empiricamente em conjunto. A Figura 1 representa essas relações como modelo interpretativo, no qual as setas indicam associações recorrentes no corpus, e não relações de causa e efeito.

Figura 1 – Modelo interpretativo das relações entre as categorias de impacto



Fonte: elaborado pelos autores (2026), com base nos estudos analisados. Nota: as setas indicam associações recorrentes na literatura analisada, e não relações causais validadas empiricamente.

4.7 RELAÇÃO COM A PERGUNTA DE PESQUISA

Considerando o corpus analisado, a literatura identifica impactos da dívida técnica tanto sobre a manutenção quanto sobre a evolução dos sistemas de software. Na manutenção, os efeitos mais consistentes estão relacionados à redução da manutenibilidade, aumento do esforço corretivo, maior propensão a defeitos, retrabalho, perda de produtividade e aumento de custos. Na evolução, aparecem dificuldades para implementar novas funcionalidades, necessidade de esforço adicional para compreender ou modificar estruturas existentes, degradação arquitetural e redução da capacidade de entrega.

A literatura também indica que esses impactos se relacionam ao surgimento e ao acúmulo da própria dívida. A pressão por entregas rápidas aparece como causa recorrente, enquanto a ausência de rastreamento sistemático e a permanência de itens durante atividades de evolução podem favorecer sua continuidade. Dessa forma, a dívida técnica apresenta-se não apenas como consequência de decisões de curto prazo, mas como um fenômeno capaz de influenciar continuamente o trabalho futuro de manutenção e evolução.

4.8 LIMITAÇÕES E ALCANCE DAS EVIDÊNCIAS

A análise comparativa revelou limitações importantes no conjunto de evidências. Parte dos trabalhos concentra-se em contextos específicos, como desenvolvimento ágil, projetos de código aberto, organizações de grande porte ou tipos particulares de dívida. Assim, resultados quantitativos como percentuais de tempo perdido ou custos adicionais devem ser interpretados como evidências do contexto estudado, e não como valores universais aplicáveis a qualquer projeto.

Também existe heterogeneidade na operacionalização da dívida técnica. Alguns estudos utilizam percepção de profissionais; outros recorrem a métricas estáticas, anomalias de modularidade, acoplamento, histórico de correções ou tarefas de manutenção. Essa diversidade amplia a visão sobre o fenômeno, mas reduz a comparabilidade direta entre resultados. Em especial, como discutido na seção 4.5, ainda não há forma uniforme de medir principal, juros e critérios de priorização (Lenarduzzi *et al.*, 2021).

Outro aspecto observado é a concentração da literatura em dívida de código e dívida arquitetural. Estudos sobre requisitos, documentação, testes e outros tipos existem, mas aparecem de forma menos consistente no corpus analisado. A dívida técnica de requisitos, por exemplo, permanece pouco explorada (Melo *et al.*, 2022). Essa assimetria significa que a força das conclusões desta pesquisa é maior para impactos associados à manutenção, arquitetura e produtividade do que para todos os tipos de dívida em igual medida.

Também foi observada recorrência de alguns autores entre os estudos centrais do corpus, especialmente em trabalhos sobre dívida arquitetural, produtividade e gerenciamento da dívida técnica. Essa concentração pode reduzir a independência entre parte das evidências e fazer com que determinadas linhas de investigação tenham maior peso na síntese. Por esse motivo, a convergência entre estudos foi

interpretada considerando não apenas o número de publicações, mas também a diversidade de métodos, contextos e grupos de autoria representados.

5 CONSIDERAÇÕES FINAIS

A pesquisa teve como objetivo analisar os impactos da dívida técnica na manutenção e evolução de sistemas de software com base na literatura acadêmica. A análise do corpus permite responder que esses impactos se manifestam em diferentes dimensões técnicas, econômicas e operacionais e tendem a se relacionar entre si ao longo do ciclo de vida do software.

Em relação à manutenção, os estudos analisados apontam efeitos recorrentes sobre a manutenibilidade, o esforço corretivo, a produtividade, o retrabalho e os custos. A dívida técnica pode estar associada a estruturas menos estáveis ou mais difíceis de modificar, a maior frequência e esforço na correção de defeitos e à necessidade de atividades adicionais de teste, análise e refatoração. Em alguns contextos, os estudos também quantificam perdas significativas de tempo de desenvolvimento e esforço adicional de manutenção.

Quanto à evolução, a literatura indica que a dívida técnica pode reduzir a capacidade de introduzir funcionalidades e realizar mudanças com eficiência, especialmente quando envolve decisões arquiteturais de grande alcance. A presença de dívida durante atividades de evolução também pode favorecer sua permanência ou a introdução de novos itens, prolongando o problema ao longo de diferentes versões do sistema.

Quanto aos tipos, a literatura apresenta múltiplos tipos de dívida técnica, incluindo requisitos, arquitetura, design, código, testes, documentação, infraestrutura e outros artefatos do desenvolvimento. Entre os fatores de surgimento, destacam-se pressões de prazo e foco em entregas rápidas, além de problemas de arquitetura e design. O acúmulo está associado à permanência de itens sem tratamento, ao rastreamento pouco sistemático e à introdução de nova dívida durante o desenvolvimento e a evolução.

A categorização realizada nesta pesquisa reuniu os impactos nas seguintes dimensões: manutenibilidade, esforço de manutenção e correção, produtividade, retrabalho, custos, degradação estrutural e capacidade de evolução e entrega. Essa organização permite observar que o impacto da dívida técnica não ocorre de forma isolada. Problemas estruturais podem aumentar a dificuldade de manutenção; o aumento do esforço pode reduzir a produtividade e elevar custos; e esses efeitos podem, por sua vez, limitar a capacidade de evolução e atrasar entregas. Essas relações, sintetizadas na Figura 1, têm caráter interpretativo, e não causal.

Como limitação, por se tratar de revisão integrativa, o levantamento não tem a exaustividade de uma revisão sistemática, e a ausência de registro uniforme do número de resultados por base impede a apresentação de um fluxo de seleção com contagens por etapa. Além disso, os estudos analisados utilizam

definições, contextos e formas de mensuração distintas para dívida técnica e seus impactos, o que limita comparações quantitativas diretas. Alguns resultados derivam da percepção de profissionais, enquanto outros são obtidos por meio de métricas, históricos de manutenção ou estudos de caso específicos. Também há recorrência de alguns autores entre trabalhos centrais do corpus, de modo que parte das evidências não pode ser tratada como inteiramente independente apenas pelo número de publicações. Consequentemente, percentuais observados em um estudo não devem ser generalizados para todos os projetos de software, e convergências entre trabalhos devem ser interpretadas considerando método, contexto e autoria. A pesquisa privilegiou estudos acadêmicos verificáveis e diretamente relacionados à pergunta de pesquisa, mantendo explícita a diferença entre a evidência reportada pelas fontes e a síntese interpretativa produzida neste trabalho.

De modo geral, a literatura analisada sustenta a conclusão de que a dívida técnica representa um fator relevante para a sustentabilidade de sistemas de software, pois seus efeitos podem aumentar o esforço necessário para manter sistemas existentes e reduzir a eficiência com que eles podem evoluir. As evidências são mais consistentes quanto ao esforço de manutenção, à produtividade e aos efeitos arquiteturais sobre mudanças futuras, enquanto outras categorias apresentam sustentação mais dependente do contexto. A compreensão desses impactos contribui para tornar mais explícitas as consequências de decisões técnicas de curto prazo e fornece base para discutir quando e por que o gerenciamento da dívida técnica se torna necessário.

Os resultados têm implicações práticas para equipes e gestores de software. Para as equipes, a associação recorrente entre pressão de prazo e surgimento da dívida e o rastreamento pouco sistemático observado no corpus indicam a importância de registrar explicitamente a dívida contraída, para que ela possa ser acompanhada e tratada. Para os gestores, a evidência de que a dívida consome parcela relevante do tempo de desenvolvimento e eleva custos de manutenção indica que ela deve ser considerada no planejamento e na priorização, ao lado de novas funcionalidades e correções. O rastreamento sistemático e o acompanhamento de indicadores estruturais, como acoplamento e densidade de dívida, podem tornar esses impactos visíveis antes que comprometam a capacidade de evolução e entrega.

Como trabalhos futuros, sugere-se: conduzir uma revisão sistemática com protocolo registrado e fluxo de seleção documentado no modelo PRISMA 2020, a fim de verificar a estabilidade das categorias aqui propostas; investigar empiricamente as relações representadas na Figura 1, por meio de estudos longitudinais que combinem métricas de projetos e percepção de profissionais; ampliar a análise para tipos de dívida menos explorados, como requisitos, testes e documentação; e aplicar a categorização em estudos de caso no contexto brasileiro, em diálogo com os resultados do projeto InsighTD.

DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL

Durante a elaboração deste artigo, foram utilizadas as ferramentas Claude 5.5, da Anthropic, e GPT 5.6 Luna e GPT 5.6 Sol, da OpenAI, para auxiliar na triagem e no fichamento de artigos, na organização das evidências, na revisão textual e na verificação de conformidade com as normas da ABNT. As sugestões das ferramentas foram conferidas pelos autores com os textos originais dos estudos e com as normas aplicáveis. O conteúdo produzido com esse auxílio foi examinado, revisado e validado pelos autores, que assumem integral responsabilidade pelas informações, análises e conclusões apresentadas no trabalho.

REFERÊNCIAS

ALVES, Nicolli S. R.; MENDES, Thiago S.; MENDONÇA, Manoel G. de; SPÍNOLA, Rodrigo O.; SHULL, Forrest; SEAMAN, Carolyn B. Identification and management of technical debt: A systematic mapping study. **Information and Software Technology**, v. 70, p. 100-121, 2016. DOI: 10.1016/j.infsof.2015.10.008.

AMANATIDIS, Theodoros; CHATZIGEORGIOU, Alexander; AMPATZOGLOU, Apostolos. The relation between technical debt and corrective maintenance in PHP web applications. **Information and Software Technology**, v. 90, p. 70-74, 2017. DOI: 10.1016/j.infsof.2017.05.004.

AVGERIOU, Paris; KRUCHTEN, Philippe; OZKAYA, Ipek; SEAMAN, Carolyn. Managing technical debt in software engineering (Dagstuhl Seminar 16162). **Dagstuhl Reports**, v. 6, n. 4, p. 110-138, 2016. DOI: 10.4230/DagRep.6.4.110.

BEHUTIYE, Woubshet Nema; RODRÍGUEZ, Pilar; OIVO, Markku; TOSUN, Ayşe. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. **Information and Software Technology**, v. 82, p. 139-158, 2017. DOI: 10.1016/j.infsof.2016.10.004.

BENNETT, Keith H.; RAJLICH, Václav T. Software maintenance and evolution: a roadmap. In: CONFERENCE ON THE FUTURE OF SOFTWARE ENGINEERING, 2000, Limerick. **Proceedings** [...]. New York: Association for Computing Machinery, 2000. p. 73-87. DOI: 10.1145/336512.336534.

BESKER, Terese; MARTINI, Antonio; BOSCH, Jan. Managing architectural technical debt: A unified model and systematic literature review. **Journal of Systems and Software**, v. 135, p. 1-16, 2018. DOI: 10.1016/j.jss.2017.09.025.

BESKER, Terese; MARTINI, Antonio; BOSCH, Jan. Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work. **Journal of Systems and Software**, v. 156, p. 41-61, 2019. DOI: 10.1016/j.jss.2019.06.004.

CALIXTO, Felipe E. de O.; ARAÚJO, Eliane C.; ALVES, Everton L. G. How does Technical Debt Evolve within Pull Requests? An Empirical Study with Apache Projects. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES), 38., 2024, Curitiba. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 212-223. DOI: 10.5753/sbes.2024.3368.

CONEJERO, José M.; RODRÍGUEZ-ECHEVERRÍA, Roberto; HERNÁNDEZ, Juan; CLEMENTE, Pedro J.; ORTIZ-CARABALLO, Carmen; JURADO, Elena; SÁNCHEZ-FIGUEROA, Fernando. Early

evaluation of technical debt impact on maintainability. **Journal of Systems and Software**, v. 142, p. 92-114, 2018. DOI: 10.1016/j.jss.2018.04.035.

CUNNINGHAM, Ward. The WyCash portfolio management system. In: CONFERENCE ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES, AND APPLICATIONS (OOPSLA), 1992, Vancouver. **Addendum to the proceedings** [...]. New York: Association for Computing Machinery, 1992. p. 29-30. DOI: 10.1145/157709.157715.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC/IEEE 14764:2022**: Software engineering — Software life cycle processes — Maintenance. 3. ed. Geneva: ISO, 2022.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 25010:2023**: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model. 2. ed. Geneva: ISO, 2023.

KITCHENHAM, Barbara; CHARTERS, Stuart. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. Version 2.3. EBSE Technical Report EBSE-2007-01. Keele: Keele University; Durham: University of Durham, 2007.

KRUCHTEN, Philippe; NORD, Robert L.; OZKAYA, Ipek. Technical Debt: From Metaphor to Theory and Practice. **IEEE Software**, v. 29, n. 6, p. 18-21, 2012. DOI: 10.1109/MS.2012.167.

LEHMAN, Meir M. Programs, life cycles, and laws of software evolution. **Proceedings of the IEEE**, v. 68, n. 9, p. 1060-1076, 1980. DOI: 10.1109/PROC.1980.11805.

LENARDUZZI, Valentina; BESKER, Terese; TAIBI, Davide; MARTINI, Antonio; ARCELLI FONTANA, Francesca. A systematic literature review on Technical Debt prioritization: strategies, processes, factors, and tools. **Journal of Systems and Software**, v. 171, art. 110827, 2021. DOI: 10.1016/j.jss.2020.110827.

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, 2015. DOI: 10.1016/j.jss.2014.12.027.

MACCORMACK, Alan; STURTEVANT, Daniel J. Technical debt and system architecture: The impact of coupling on defect-related activity. **Journal of Systems and Software**, v. 120, p. 170-182, 2016. DOI: 10.1016/j.jss.2016.06.007.

MARTINI, Antonio; BESKER, Terese; BOSCH, Jan. Technical Debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. **Science of Computer Programming**, v. 163, p. 42-61, 2018. DOI: 10.1016/j.scico.2018.03.007.

MELO, Ana; FAGUNDES, Roberta; LENARDUZZI, Valentina; SANTOS, Wylliams Barbosa. Identification and measurement of Requirements Technical Debt in software development: A systematic literature review. **Journal of Systems and Software**, v. 194, art. 111483, 2022. DOI: 10.1016/j.jss.2022.111483.

MENDES, Thiago Souto; FARIAS, Mário André de Freitas; MENDONÇA, Manoel G.; SOARES, Henrique Frota; KALINOWSKI, Marcos; SPÍNOLA, Rodrigo Oliveira. Impacts of agile requirements documentation debt on software projects: a retrospective study. In: ANNUAL ACM SYMPOSIUM ON

APPLIED COMPUTING, 31., 2016, Pisa. **Proceedings** [...]. New York: Association for Computing Machinery, 2016. p. 1290-1295. DOI: 10.1145/2851613.2851761.

PAUDEL, Bhuwan; GONZALEZ-HUERTA, Javier; ZABARDAST, Ehsan. Exploring the evolution of technical debt in monolithic and hybrid microservice architecture: An industrial case study. **Journal of Systems and Software**, v. 237, art. 112831, 2026. DOI: 10.1016/j.jss.2026.112831.

RAMAČ, Robert; MANDIĆ, Vladimir; TAUŠAN, Nebojša; RIOS, Nicolli; FREIRE, Sávio; PÉREZ, Boris; CASTELLANOS, Camilo; CORREAL, Darío; PACHECO, Alexia; LOPEZ, Gustavo; IZURIETA, Clemente; SEAMAN, Carolyn; SPÍNOLA, Rodrigo. 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, art. 111114, 2022. DOI: 10.1016/j.jss.2021.111114.

RIOS, Nicolli; MENDONÇA NETO, Manoel Gomes de; SPÍNOLA, Rodrigo Oliveira. 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. DOI: 10.1016/j.infsof.2018.05.010.

RIOS, Nicolli; SPÍNOLA, Rodrigo Oliveira; MENDONÇA, Manoel; SEAMAN, Carolyn. The most common causes and effects of technical debt: first results from a global family of industrial surveys. In: ACM/IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT (ESEM), 12., 2018, Oulu. **Proceedings** [...]. New York: Association for Computing Machinery, 2018. art. 39, p. 1-10. DOI: 10.1145/3239235.3268917.

RIOS, Nicolli; SPÍNOLA, Rodrigo Oliveira; MENDONÇA, Manoel; SEAMAN, Carolyn. 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, 2020. DOI: 10.1007/s10664-020-09832-9.

TOM, Edith; AURUM, Aybüke; VIDGEN, Richard T. An exploration of technical debt. **Journal of Systems and Software**, v. 86, n. 6, p. 1498-1516, 2013. DOI: 10.1016/j.jss.2012.12.052.

VERDECCHIA, Roberto; KRUCHTEN, Philippe; LAGO, Patricia; MALAVOLTA, Ivano. Building and evaluating a theory of architectural technical debt in software-intensive systems. **Journal of Systems and Software**, v. 176, art. 110925, 2021. DOI: 10.1016/j.jss.2021.110925.

WOHLIN, Claes. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING (EASE), 18., 2014, London. **Proceedings** [...]. New York: Association for Computing Machinery, 2014. art. 38, p. 1-10. DOI: 10.1145/2601248.2601268.

WOLFART, Daniele; MARTINEZ, Jabier; ASSUNÇÃO, Wesley K. G.; COLANZI, Thelma E.; EGYED, Alexander. Variability debt in opportunistic reuse: A multi-project field study. **Journal of Systems and Software**, v. 210, art. 111969, 2024. DOI: 10.1016/j.jss.2024.111969.