UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO INTERDISCIPLINAR EM TECNOLOGIA DA INFORMAÇÃO

PEDRO EMANUEL MAIA DE MESQUITA

SOBRECARGA DE DOCUMENTAÇÃO EM PROJETOS DE SOFTWARE: UMA REVISÃO SISTEMÁTICA DA LITERATURA

PAU DOS FERROS
2025

PEDRO EMANUEL MAIA DE MESQUITA

SOBRECARGA DE DOCUMENTAÇÃO EM PROJETOS DE SOFTWARE: UMA REVISÃO SISTEMÁTICA DA LITERATURA

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel Interdisciplinar em Tecnologia da Informação.

Orientador: Reudisman Rolim de Sousa, Prof. Dr.

PAU DOS FERROS

2025

M578s Mesquita, Pedro Emanuel Maia de .
Sobrecarga de documentação em projetos de software: uma revisão sistemática da literatura / Pedro Emanuel Maia de Mesquita. - 2025.
37 f. : il.

Orientador: Reudisman Rolim de Sousa.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2025.

1. Engenharia de software. 2. Sobrecarga documental. 3. Documentação mínima viável. 4. Revisão sistemática da literatura. 5. Agilidade. I. Sousa, Reudisman Rolim de , orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

PEDRO EMANUEL MAIA DE MESQUITA

SOBRECARGA DE DOCUMENTAÇÃO EM PROJETOS DE SOFTWARE: UMA REVISÃO SISTEMÁTICA DA LITERATURA

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel Interdisciplinar em Tecnologia da Informação.

Defendida em: 10 / 12 / 2025

BANCA EXAMINADORA

Reudisman Rolim de Sousa, Prof. Dr. (UFERSA)
Presidente

Bruno Borges da Silva, Prof. Me. (UFERSA)
Membro Examinador

George Felipe Fernandes Vieira, Prof. Me. (UFERSA)
Membro Examinador

AGRADECIMENTOS

Agradeço primeiramente à minha família pelo amor incondicional, pelo apoio constante e pela paciência nos momentos mais desafiadores desta jornada. Cada palavra de incentivo, cada gesto de compreensão e cada demonstração de carinho foram fundamentais para que eu pudesse seguir em frente. À minha mãe, por ter me ensinado o valor da educação e por ser meu exemplo como pessoa; às minhas irmãs e ao meu pai, por acreditarem em mim mesmo quando duvidei das minhas próprias capacidades, e por me ensinarem o valor do esforço, da honestidade e da perseverança.

Aos meus amigos e aqueles que fiz e conheci ao longo do meu percurso acadêmico, que estiveram ao meu lado, compartilhando risadas, conselhos e apoio em cada etapa desta caminhada acadêmica. A amizade de vocês tornou o percurso mais leve e repleto de aprendizados. Cada conversa, cada troca e cada momento de descontração contribuíram para que eu mantivesse o equilíbrio e a motivação para concluir este trabalho.

Ao meu orientador, professor Reudismam Rolim de Sousa, expresso minha mais profunda gratidão pela dedicação, paciência e orientação. Seu comprometimento acadêmico e humano foi essencial para o desenvolvimento deste trabalho. Agradeço por ter me guiado com sabedoria, por suas contribuições valiosas e por ter me incentivado a buscar sempre o melhor de mim, tanto como aluno quanto como pessoa.

Por fim, agradeço a todos que, de alguma forma, fizeram parte dessa trajetória, sejam colegas, professores e profissionais que contribuíram direta ou indiretamente para a minha formação. A realização deste Trabalho de Conclusão de Curso é fruto de um conjunto de esforços, e a cada um que me apoiou, deixo registrada minha sincera gratidão.

EPÍGRAFE

Existem duas maneiras de construir um projeto de software: Uma é torná-lo tão simples que, ob- viamente, não há deficiências, e a outra é torná- lo tão complicado que não há deficiências óbvias. O primeiro método é muito mais difícil.

Charles Antony Richard Hoare

RESUMO

A documentação de software constitui um elemento fundamental para o sucesso de projetos de desenvolvimento, manutenção e evolução de sistemas. Entretanto, observa-se que o excesso de artefatos documentais, muitas vezes produzidos sem critérios claros de relevância e atualização, gera a chamada sobrecarga documental, fenômeno que impacta negativamente a produtividade, os prazos e a qualidade dos produtos de software. Este trabalho tem como objetivo analisar os impactos da sobrecarga documental na Engenharia de Software, identificando suas principais causas, consequências e possíveis estratégias de mitigação. Para isso, foi conduzida uma Revisão Sistemática da Literatura (RSL), fundamentada nas diretrizes de Kitchenham e Charters (2007), abrangendo publicações recentes das principais bases de dados científicas. O estudo examina a sobrecarga documental sob perspectivas técnicas, humanas e organizacionais, abordando desde as práticas tradicionais de engenharia de requisitos até os desafios da documentação em contextos ágeis, colaborativos, *low-code* e de software livre. Os resultados evidenciam que a sobrecarga documental não decorre apenas do volume de informação, mas também de problemas semânticos, falta de padronização, redundância e ausência de automação. Com base na análise dos trabalhos, propõe-se um *framework* conceitual de documentação mínima viável, orientado à clareza, rastreabilidade e valor agregado, capaz de equilibrar o rigor técnico com a agilidade requerida pelos ambientes de desenvolvimento modernos.

Palavras-chave: engenharia de software; sobrecarga documental; documentação mínima viável; revisão sistemática da literatura; agilidade.

ABSTRACT

Software documentation is a fundamental element for the success of software development, main- tenance, and evolution projects. However, the excessive production of documentation artifacts often created without clear criteria for relevance and updating leads to documentation overload, a phenomenon that negatively affects productivity, deadlines, and software quality. This study aims to analyze the impacts of documentation overload in Software Engineering, identifying its main causes, consequences, and mitigation strategies. To achieve this, a Systematic Literature Review (SLR) was conducted following the guidelines of Kitchenham and Charters (2007), covering recent publications from major scientific databases. The research examines documen- tation overload from technical, human, and organizational perspectives, addressing traditional requirements engineering practices as well as challenges in agile, collaborative, low-code, and open-source contexts. The findings indicate that documentation overload stems not only from the volume of artifacts but also from semantic inconsistencies, lack of standardization, redundancy, and insufficient automation. Based on the analyzed literature, the study proposes a conceptual framework for minimum viable documentation, aimed at achieving clarity, traceability, and value while balancing technical rigor with the agility demanded by modern software development environments.

Keywords: software engineering; documentation overload; minimal viable documentation; systematic literature review; agility.

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A documentação desempenha um papel central no desenvolvimento de software, servindo como meio de comunicação, registro de decisões e referência técnica para todos os *stakeholders* envolvidos no projeto. Desde a concepção inicial até a manutenção e evolução do sistema, os documentos atuam como elementos estruturantes que garantem rastreabilidade, clareza e padronização no processo de engenharia de software. Entretanto, nas últimas décadas, observa-se um crescente debate acerca do equilíbrio entre o nível de documentação exigido e a agilidade necessária para a entrega contínua de valor, o que tem levado ao fenômeno conhecido como sobrecarga documental, impactando a eficiência, os prazos e a qualidade de projetos modernos. A literatura clássica em Engenharia de Software destaca que documentos excessivamente extensos geram a chamada "ilusão de controle", mascarando riscos dinâmicos e aumentando o retrabalho (Sommerville, 2019). Como os requisitos são intrinsecamente voláteis, a manutenção de grandes volumes documentais rapidamente se torna onerosa e improdutiva. Assim, quando a documentação passa a exigir mais esforço do que o próprio desenvolvimento, ela perde a função

estratégica e se transforma em um obstáculo operacional.

Esse problema é frequente em organizações que seguem rigorosamente modelos de qualidade e conformidade, como o CMMI e a ISO/IEC 25010. Tais modelos, embora fundamentais, podem gerar burocracia documental, resultando em atrasos e desmotivação das equipes. Conforme apontado por Poppendieck e Poppendieck (2003), a burocracia é um inimigo silencioso da inovação, pois consome recursos sem agregar valor ao produto final.

Com o advento do Manifesto Ágil (Beck *et al.*, 2001), o setor passou a priorizar software funcionando em vez de documentação abrangente, impulsionando abordagens mais leves, colabo- rativas e iterativas. A mudança não propõe o abandono da documentação, mas sua reinterpretação: documentar apenas o que é necessário para garantir comunicação, entendimento e rastreabilidade. Isso é reforçado por Patton (2014), com o conceito de *Just Enough Documentation*, que direciona esforços a artefatos com valor claro e mensurável.

Hoy e Xu (2023) demonstram que, embora a agilidade traga ganhos significativos, ainda há desafios relacionados à documentação mínima e à rastreabilidade em ambientes de grande escala. Lent (2025) acrescenta que metodologias ágeis mal aplicadas podem levar ao outro extremo: escassez documental, causando perdas de conhecimento e inconsistência técnica.

Além dos desafios metodológicos, práticas estruturadas de investigação científica, como a RSL — Revisão Sistemática da Literatura (Kitchenham; Charters, 2007; Klock, 2021),

demonstram que a documentação desempenha um papel essencial para a transparência, reprodutibilidade e análise histórica, princípios igualmente válidos em contextos corporativos.

Casos emblemáticos, como a falha do portal healthcare.gov nos EUA, reforçam que a ausência de integração e gestão documental adequada pode resultar em prejuízos de grande escala, evidenciando a necessidade de governança eficiente sobre artefatos técnicos (GAO, 2014). Sob outra perspectiva, Coelho (2009) destaca que a falta de padronização documental compromete a preservação do conhecimento organizacional, tornando os sistemas dependentes exclusivamente da memória tácita e aumentando os custos de manutenção.

Portanto, torna-se evidente que a solução não está em eliminar a documentação, mas em equilibrar rigor e agilidade, garantindo que cada documento produzido tenha relevância efetiva para o avanço do projeto. Nesse sentido, compreender o impacto da sobrecarga documental é cru- cial, não apenas por sua recorrência na indústria, mas pelos efeitos diretos sobre a produtividade, a motivação das equipes e o valor entregue ao cliente.

Diante disso, o presente trabalho propõe uma RSL sobre a sobrecarga documental em projetos de software, com o objetivo de identificar as principais causas, consequências e estratégias mitigadoras reportadas na literatura científica. A metodologia adotada segue as diretrizes de Kitchenham e Charters (2007), amplamente reconhecidos na Engenharia de Software, assegurando rigor, reprodutibilidade e imparcialidade na coleta e análise dos estudos.

Ao sintetizar os achados da literatura, este estudo pretende contribuir para a construção de uma base conceitual sólida que auxilie pesquisadores e profissionais na formulação de práticas equilibradas de documentação. Espera-se, assim, oferecer subsídios para a criação de um *framework* de documentação mínima viável, que preserve a rastreabilidade e o controle, sem sacrificar a agilidade e a inovação, fatores essenciais para a competitividade e a sustentabilidade de projetos de software contemporâneos.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica deste trabalho busca consolidar os principais conceitos, abordagens e evidências científicas que sustentam a discussão sobre a sobrecarga de documentação em projetos de software. O tema é multifacetado e envolve dimensões técnicas, humanas, organizacionais e históricas da Engenharia de Software. Assim, esta seção está estruturada em blocos temáticos que percorrem os seguintes eixos: (i) Fundamentos da Engenharia de Software e Processos de Desenvolvimento; (ii) paradigmas ágeis, práticas contemporâneas e dimensões sociotécnicas da documentação; (iii) documentação como artefato técnico, comunicacional e organizacional; e (iv) Qualidade da Informação e suas Implicações no Desenvolvimento de Software.

2.1 Fundamentos da engenharia de software e processos de desenvolvimento

O desenvolvimento de software depende de processos bem estruturados que permitam gerenciar a complexidade crescente dos sistemas e garantir a comunicação eficaz entre os *stakeholders* envolvidos. Desde as etapas iniciais de concepção até a entrega e manutenção do produto, a documentação atua como suporte para decisões técnicas, registro de requisitos e memória organizacional, sendo a base para a rastreabilidade e continuidade do projeto.

A etapa de elicitação de requisitos, conforme apresentado por Gomes (2000), depende da clareza comunicacional para que as necessidades reais do usuário sejam corretamente compreendidas e traduzidas em artefatos técnicos. Quando tal processo carece de objetividade e foco, tende a gerar excesso de formalização, resultando em documentação redundante ou pouco útil.

Nesse sentido, Debastiani (2016) destaca que a definição de escopo é fundamental para evitar que a produção documental ultrapasse o que é realmente necessário para o projeto. A falta desse alinhamento pode causar sobrecarga documental, uma vez que documentos superdimensionados são criados na tentativa de compensar lacunas de entendimento ou inseguranças gerenciais.

A necessidade de uma produção documental equilibrada também é reforçada por Martins (2007), que ressalta que o gerenciamento de projetos deve manter um equilíbrio entre tempo, custo e qualidade, de modo que a documentação seja um instrumento de apoio ao desenvolvimento e não um elemento que acarrete atrasos ou retrabalho.

Complementando essa visão, Silva (2020) destaca as diferenças entre a documentação tradicional, geralmente extensa, estática e suscetível à obsolescência, e a *living documentation*,

que acompanha a evolução do software em tempo real, por meio da automação e integração direta com o código. Essa abordagem reduz desperdícios e evita que artefatos se tornem irrelevantes com o passar do tempo, contribuindo diretamente para mitigar a sobrecarga documental.

Reis (2003), ao analisarem processos de desenvolvimento em software livre, observam que, embora haja uma maior informalidade nas práticas de produção documental, ainda assim, mínimos estruturais são necessários para garantir o alinhamento entre colaboradores distribuídos. Essa visão é complementada por Garofalo e Alis (2010), que discutem a documentação como meio de democratização do acesso ao conhecimento nesses projetos, ao mesmo tempo em que alertam para barreiras quando a informação é insuficiente ou apresentada de forma inadequada.

De forma semelhante, Pinho (2024) identifica que, em projetos colaborativos de código aberto, os principais desafios estão relacionados à falta de padronização e à atualização contínua dos artefatos documentais, o que prejudica sua utilidade prática e conduz à sobrecarga por documentação obsoleta ou não rastreável.

A relevância da documentação para a manutenção é profundamente discutida por Bernardo (2022), que explora os problemas mais recorrentes, destacando erros semânticos, inconsistências terminológicas e acúmulo de artefatos sem uso. Segundo os autores, a qualidade documental deve estar alinhada ao propósito do sistema; caso contrário, o excesso informacional torna-se prejudicial ao ciclo de desenvolvimento.

Nas metodologias de desenvolvimento contemporâneo, como o Scrum, a documentação é tratada como um artefato leve e objetivo, voltado à comunicação clara e à inspeção contínua. Silva e Lovato (2016) demonstram que o *framework* contribui para a eliminação de burocracias, priorizando o valor entregue ao cliente e reduzindo a produção documental sem finalidade.

Por fim, o estudo de Carlos e Haddad (2025) evidencia que, em equipes *low-code*, mesmo com desenvolvimento acelerado e visual, a documentação permanece essencial para assegurar rastreabilidade e padronização, especialmente em ambientes nos quais não há acesso direto ao código-fonte. Dessa forma, novas tecnologias não eliminam a necessidade documental, apenas alteram sua forma, escopo e dinâmica.

Diante disso, compreende-se que a documentação funciona como elemento estruturante da Engenharia de Software; porém, sua produção deve ser racional e proporcional à necessidade real do projeto. Quando esse equilíbrio não é observado, emerge o fenômeno da sobrecarga documental, que conduz à perda de eficiência e compromete a manutenção e evolução dos sistemas.

2.2 Paradigmas ágeis, práticas contemporâneas e dimensões sociotécnicas da documentação

Com a consolidação das metodologias ágeis e da colaboração em escala global, a documentação de software passou a ser compreendida não apenas como um artefato técnico, mas também como um elemento sociotécnico, influenciado por fatores humanos, organizacionais e culturais. Nesse contexto, pesquisas recentes têm buscado entender como os desenvolvedo- res percebem, produzem e utilizam documentação em ambientes modernos de engenharia de software.

Arya, Guo e Robillard (2024) investigam por que desenvolvedores contribuem com documentação, concluindo que aspectos como reconhecimento profissional, utilidade comunitária e motivação intrínseca influenciam diretamente a qualidade dos registros técnicos. Esse componente emocional também é analisado por Venigalla e Chimalakonda (2021), que identificaram sentimentos de frustração e desvalorização associados à atividade de documentar, especialmente quando as organizações não reconhecem formalmente a importância desse trabalho. Esses fatores contribuem para a despriorização da documentação, levando à desatualização e à sobrecarga informacional posterior.

Em ambientes ágeis, Behutiye *et al.* (2020) apontam que os requisitos de qualidade (como desempenho, segurança e usabilidade) são frequentemente documentados de maneira superficial, já que o foco principal permanece nos requisitos funcionais. Da mesma forma, Nasir *et al.* (2023) evidenciam que equipes ágeis ainda têm dificuldades em registrar requisitos não funcionais, o que gera lacunas críticas no entendimento do produto e aumenta a dependência de conhecimento tácito, um gatilho direto para o acúmulo documental tardio.

A ausência de padrões efetivos também é uma preocupação. Santos e Correia (2020) realizaram uma revisão sobre linguagens de padrões para documentação, defendendo que a padronização textual e visual dos artefatos melhora a compreensão entre *stakeholders* e reduz a ambiguidade interpretativa. Essa perspectiva é complementada por Nassif e Robillard (2025), que analisam formas de apresentação da documentação e mostram que formatos inadequados impactam negativamente o uso e a manutenção dos documentos.

Outro desafio crítico abordado pela literatura atual é o *drift documental*, ou desalinhamento entre documentação, *design* e implementação. Romeo *et al.* (2024) demonstram que essa divergência costuma ocorrer quando a documentação não acompanha a evolução do software devido à falta de automação e integração contínua com o código, gerando um fenômeno que resulta em sobrecarga acumulativa e progressiva perda de confiabilidade dos artefatos.

Nesse sentido, ferramentas automatizadas têm sido alternativas promissoras. O trabalho de Franca *et al.* (2023) apresenta o LinkDoc, um processo automatizado para apoiar a entrega de documentação em ambientes de desenvolvimento distribuído, reduzindo o esforço manual e os erros de sincronização. A automação também é abordada por Tileria, Blasco e Dash (2024) com o DocFlow, uma ferramenta capaz de extrair especificações de segurança a partir de documentação textual, auxiliando equipes a manter a conformidade sem ampliar o volume de artefatos produzidos manualmente.

Além disso, Sovrano, Lognoul e Bacchelli (2024) discutem a transparência na documentação de plataformas *online*, especialmente no contexto regulatório europeu, demonstrando que a conformidade normativa exige documentação clara e auditável, porém sem excessos que comprometam a agilidade. Nesse cenário, a burocracia imposta pelas regulações pode fomentar a sobrecarga documental se não houver estratégias de sintetização e automatização.

2.3 Documentação como artefato técnico, comunicacional e organizacional

A documentação de software possui um papel estruturante no ciclo de vida do desenvolvimento, funcionando como meio de comunicação, preservação do conhecimento e suporte à manutenção. Yamanouchi (2020) destaca que a ausência ou deficiência documental compromete a compreensão do sistema, tornando as equipes dependentes do conhecimento tácito e aumentando os custos de evolução. Essa perspectiva reforça a importância da documentação como instrumento de continuidade e qualidade do produto.

No contexto da manutenção, etapa responsável pela maior parte do custo de um sistema ao longo do tempo, a documentação torna-se ainda mais crítica. Souza *et al.* (2004a) e Souza *et al.* (2004b) demonstram que documentos bem estruturados facilitam correções, melhorias e adaptações, ao passo que a documentação desatualizada resulta em retrabalho, inconsistência e risco operacional. Esses efeitos estão diretamente associados ao fenômeno da sobrecarga documental acumulativa, em que artefatos volumosos e redundantes perdem utilidade e confiança ao longo do tempo. Embora estudos clássicos, como os de Souza et al. (2004a, 2004b) e Sandhof e Filgueiras (2006), tenham estabelecido a importância da documentação como elemento crítico para a fase de manutenção, o contexto contemporâneo do desenvolvimento de software apresenta desafios significativamente distintos.

Com a consolidação de práticas como DevOps, integração e entrega contínuas (CI/CD) e arquiteturas baseadas em microsserviços, a manutenção deixa de ser uma etapa isolada e passa a ocorrer de forma contínua ao longo do ciclo de vida do software. Nesse cenário, a documentação

precisa acompanhar mudanças frequentes, rápidas e incrementais, reforçando o conceito de *living documentation*.

Estudos mais recentes, como o de Theunissen, Heesch e Avgeriou (2022), evidenciam que, em ambientes de desenvolvimento contínuo, a documentação assume o papel de artefato vivo, sendo atualizada de forma automatizada e integrada aos pipelines de CI/CD. De forma complementar, Romeo *et al.* (2024) demonstram que a falta de sincronização entre código e documentação gera o fenômeno conhecido como *documentation drift*, impactando diretamente a eficiência da manutenção e a compreensão do sistema.

Assim, a manutenção moderna exige abordagens documentais dinâmicas, automatizadas e orientadas à evolução contínua do software, ampliando e atualizando as perspectivas apresentadas pelos estudos clássicos.

Outro aspecto essencial refere-se à linguagem e à clareza textual. Monfardini (2025) argumenta que o redator técnico deve garantir a consistência terminológica, a objetividade e a padronização, pois documentos complexos ou mal escritos contribuem para erros de interpretação. Essa observação converge com Sandhof e Filgueiras (2006), que identificam que diversos defeitos em software têm origem em falhas humanas de entendimento, frequentemente relacionadas à comunicação ambígua ou incompleta.

Organizações modernas também precisam lidar com o controle de versões da documentação, de modo similar ao que já é prática consolidada para código-fonte. Konnorate *et al.* (2019) mostram que o versionamento documental permite rastrear decisões e mitigar problemas de sincronização, reduzindo retrabalhos e conflitos de informação. Essa prática é especialmente importante em aplicações web, que passam por mudanças rápidas e contínuas, exigindo documentação dinâmica e constantemente atualizada (Zang; Denardi, 2005).

Pesquisas voltadas para perfis específicos de usuários reforçam a necessidade de documentação direcionada e contextualizada. Santos (2025), ao avaliar a documentação de bibliotecas ORM, destaca que a usabilidade documental impacta diretamente a produtividade dos desen- volvedores e a adoção das ferramentas. A falta de objetividade ou de exemplos práticos eleva a carga cognitiva do usuário e desencoraja a consulta formal aos documentos.

Mendes (2007) relaciona déficits de documentação à falta de rastreabilidade em erros e configurações, o que leva ao aumento da dívida técnica e à perda gradual do controle sobre o produto. Nesse sentido, Santos (2021) propõe um processo estruturado de validação semântica dos requisitos, visando reduzir ambiguidades desde as etapas iniciais e prevenindo sobrecarga documental futura.

A adaptação da documentação à realidade organizacional também é um tema recorrente

na literatura. Brandão (2019) afirma que micro e pequenas empresas tendem a negligenciar a documentação por limitações de recursos, resultando em informalidade que compromete a manutenção e a escalabilidade. Já Marques (2021) mostra que, em contextos científicos como sistemas bioinformáticos, a documentação é essencial para garantir a reprodutibilidade experimental, ampliando não apenas o valor técnico, mas também o valor científico do software. Por fim, Lacerda (2005) antecipa tendências modernas ao propor o FrameworkDoc, uma ferramenta para automação e geração de artefatos documentais, reduzindo o esforço manual e promovendo o alinhamento entre a documentação e o código. Essa linha de pesquisa aponta para uma transformação no papel da documentação: de passiva e estática para automática, rastreável e integrada ao ciclo de desenvolvimento.

Dessa forma, observa-se que a documentação, quando inadequadamente planejada ou mal executada, pode gerar acúmulo de informações desnecessárias, redução da produtividade e fragilização do conhecimento organizacional. Porém, quando planejada com propósito, mantida atualizada e automatizada, torna-se um elemento chave para a qualidade, manutenção e longevidade do software.

2.4 Qualidade da informação e suas implicações no desenvolvimento de software

A produção documental está diretamente relacionada à qualidade do software e ao desempenho organizacional. Zhi *et al.* (2015) demonstram que a documentação produz custos relevantes ao longo do ciclo de vida do software, mas tais custos são compensados quando os artefatos são úteis, acessíveis e mantidos atualizados, reduzindo retrabalho e falhas interpretativas. Quando o volume de documentação cresce sem controle ou propósito, configura-se o fenômeno da sobrecarga documental, que compromete a eficiência das equipes e dificulta o acesso a informações cruciais.

Venigalla e Chimalakonda (2024) analisam artefatos hospedados no GitHub e evidenciam que os desenvolvedores frequentemente optam por formatos mínimos ou informais de documentação, priorizando a velocidade de entrega. No entanto, a ausência de estrutura e padronização contribui para a baixa qualidade documental e para dificuldades de colaboração, especialmente em equipes distribuídas.

A qualidade da documentação também é descrita como um fator estratégico para o desempenho organizacional. Nabot e Al-Qerem (2025) destacam que a qualidade do software e a qualidade documental estão diretamente conectadas à *performance* organizacional, afetando indicadores como produtividade, retrabalho e satisfação do cliente. Na mesma linha, Khalid *et*

al. (2025) comparam abordagens de engenharia de requisitos e concluem que técnicas mais rigorosas de documentação de requisitos garantem melhor conformidade, reduzindo problemas posteriores de entendimento e implementação.

No contexto ágil, Behutiye *et al.* (2022) identificam que equipes frequentemente negligenciam requisitos de qualidade (não funcionais), o que gera lacunas significativas na rastreabilidade e deixa as equipes vulneráveis a erros arquiteturais. Essa carência tende a gerar documentação adicional tardia, contribuindo para a sobrecarga acumulativa. Complementarmente, Kubiaczyk e Jarzębowicz (2025) mostram que equipes ágeis buscam soluções híbridas que equilibram flexibilidade e formalização, mas ainda carecem de práticas claras sobre como documentar tais requisitos sem comprometer a agilidade.

O controle da qualidade textual também é necessário: desde trabalhos clássicos como Lehner (1993), já se reconhece que a compreensibilidade documental é um fator crítico, devendo ser mensurada e acompanhada. Documentação extensa e mal estruturada tende a gerar confusão, erros cognitivos e baixa reusabilidade dos artefatos.

Soluções modernas para mitigar a sobrecarga vêm sendo investigadas com foco em automação e integração contínua. Theunissen, Heesch e Avgeriou (2022) demonstram que, no desenvolvimento contínuo, a documentação deve estar integrada às pipelines de entrega, acompanhando o ritmo do código. Naimi *et al.* (2024) reforçam esse direcionamento ao utilizarem Grandes Modelos de Linguagem (LLMs) para automatizar descrições de casos de uso, reduzindo o esforço manual sem sacrificar a precisão.

Outro fator determinante para a eficiência documental relaciona-se ao estilo adotado. Nurlanuly, Araz e Zhuldiz (2025) mostram que estilos de documentação mais padronizados promovem uma melhor colaboração entre equipes, enquanto estilos não convencionais elevam a carga cognitiva de leitura e compreensão, aumentando a probabilidade de falhas humanas.

Alzahrani (2025) propõe a abordagem PDA-SDD (*Pre-During-After Software Development Documentation*), estruturando artefatos de acordo com as fases do desenvolvimento e garantindo que a produção documental não apenas acompanhe o projeto, mas também evite o acúmulo desnecessário e a desatualização.

3 METODOLOGIA

A presente pesquisa emprega uma Revisão Sistemática da Literatura (RSL) como estratégia metodológica, com o objetivo de identificar, organizar e sintetizar evidências científicas sobre a sobrecarga documental no desenvolvimento de software. O método segue as diretrizes propostas por Kitchenham e Charters (2007) para RSL em Engenharia de Software e está estruturado segundo o protocolo PRISMA (*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*), garantindo rigor metodológico, rastreabilidade e reprodutibilidade.

3.1 Objetivo da RSL

O objetivo da Revisão Sistemática é responder à seguinte pergunta de pesquisa (*Research Question* - RQ):

RQ1: Como a sobrecarga documental impacta os processos de engenharia de software e quais estratégias têm sido propostas para mitigá-la?

Para aprofundar a análise, elaboraram-se as perguntas secundárias:

RQ1.1: Como ocorre a distribuição temporal e a relevância dos estudos sobre a sobrecarga de documentação no desenvolvimento de software?

RQ1.2: Quais são os principais problemas apontados sobre a sobrecarga de documentação no desenvolvimento de software?

RQ1.3: Quais são os benefícios da adoção da documentação adequada no desenvolvimento de software?

RQ1.4: Como ocorre a documentação de software em ambientes de desenvolvimento de software ágeis?

RQ1.5: Quais são as novas tendências de automação e tecnologias no desenvolvimento de software, no tocante à documentação de software?

3.2 Estratégia de busca

A busca foi realizada em bases científicas amplamente utilizadas na área de Engenharia de Software, como IEEE Xplore, ACM Digital Library, Springer Link, ScienceDirect, Scopus e Google Scholar, como apoio complementar.

As strings de busca consideraram variações semânticas do tema, tais como:

(“software documentation” OR “technical documentation”) AND (“overload” OR “ex-

cess” OR “quality”).

(“documentation” AND “agile” AND “quality” AND “requirements”). (“documentation” AND “automation” OR “DevOps” OR “continuous software development”).

O período-alvo dos estudos da pesquisa abrange 2000 a 2025, acompanhando a adoção das metodologias ágeis e as evoluções contemporâneas.

3.3 Seleção e quantificação dos estudos

A busca sistemática resultou, inicialmente, na identificação de 114 estudos científicos provenientes de diferentes bases de dados da área de Engenharia de Software. Esses estudos passaram por todas as etapas do fluxo PRISMA: identificação, triagem, elegibilidade e inclusão, com a aplicação de critérios objetivos de qualidade, relevância e aderência ao tema. Apenas os estudos que atenderam aos critérios estabelecidos foram considerados na síntese final, totalizando 51 estudos.

Para assegurar a relevância e a qualidade, foram definidos os seguintes critérios (Tabela 1):

Tabela 1 – Critérios de inclusão e exclusão dos estudos

Tipo	Descrição
Critérios de Inclusão	• Estudos publicados entre 2000 e 2025; • Trabalhos revisados por pares e literatura cinzenta; • Estudos relacionados à documentação de software, sobrecarga documental, qualidade da documentação ou automação voltada à documentação; • Disponibilidade do texto completo.
Critérios de Exclusão	• Documentação fora do contexto de software; • Trabalhos puramente opinativos sem método definido; • Artigos duplicados entre bases; • Falta de acesso ao texto integral.

Fonte: Autoria própria (2025).

3.4 Classificação e extração dos dados

Os dados foram classificados seguindo os critérios de inclusão e exclusão, assim como foi feita uma análise da qualidade do trabalho. Os dados coletados dos trabalhos foram os seguintes: referência, ano, tipo de estudo, contribuição principal do trabalho e qualidade.

A avaliação da qualidade dos estudos selecionados foi conduzida com o objetivo de assegurar a confiabilidade, a relevância científica e a robustez metodológica dos trabalhos

incluídos nesta revisão, conforme recomendado por (Kitchenham; Charters, 2007).

Para isso, foi adotado um checklist de critérios de qualidade, baseado em práticas consolidadas em Revisões Sistemáticas da área de Engenharia de Software. Cada estudo foi avaliado individualmente a partir dos seguintes critérios:

• **CQ1**: O estudo apresenta objetivos de pesquisa claros e bem definidos?

• **CQ2**: A metodologia adotada é descrita de forma adequada e reproduzível?

• **CQ3**: O estudo apresenta fundamentação teórica consistente e alinhada ao tema da documentação de software?

• **CQ4**: Os resultados são claramente apresentados e discutidos de forma crítica?

• **CQ5**: O estudo demonstra relevância para a problemática da sobrecarga documental ou qualidade da documentação?

Cada critério foi avaliado de forma binária, recebendo pontuação *1* quando atendido e *0* quando não atendido, totalizando uma pontuação máxima de 5 pontos por estudo.

A classificação final da qualidade dos estudos foi definida a partir da pontuação total obtida no *checklist*, conforme os seguintes intervalos:

• **Alta qualidade**: estudos com pontuação entre 4 e 5;

• **Média qualidade**: estudos com pontuação igual a 3;

• **Média/Alta qualidade**: estudos com pontuação intermediária ou que apresentaram peque- nas limitações metodológicas sem comprometer a validade dos resultados.

Essa abordagem permitiu reduzir a subjetividade na avaliação, garantindo maior transpa- rência e rigor metodológico ao processo de seleção e análise dos estudos incluídos.

4 RESULTADOS

A aplicação do protocolo PRISMA, aliada às diretrizes propostas por Kitchenham e Charters (2007), resultou em uma seleção final de 51 estudos diretamente relacionados ao objetivo desta pesquisa: compreender a documentação de software sob múltiplas perspectivas, incluindo a qualidade, a redução de sobrecarga documental, a automação, padrões e o papel dentro de equipes de desenvolvimento. A análise sistemática permitiu identificar padrões consistentes entre os estudos, assim como lacunas ainda presentes na literatura.

Na Tabela 2, que se encontra no apêndice, pode ser vista uma síntese de todos os estudos inclusivos para a nossa análise.

4.1 Distribuição temporal e relevância crescente do tema

Os resultados demonstram que o interesse pela documentação de software tem crescido de forma contínua ao longo dos últimos anos. Embora estudos fundamentais remontem à década de 1990 (ex.: Lehner (1993)), o volume de publicações aumentou significativamente a partir de 2015, impulsionado por três fatores principais:

- A ascensão de práticas ágeis e o conflito entre agilidade e documentação extensiva ((Beck
- *et al.*, 2001; Patton, 2014; Poppendieck; Poppendieck, 2003).
- A intensificação da colaboração global e do desenvolvimento aberto (Pinho, 2024; Arya; Guo; Robillard, 2024; Sovrano; Lognoul; Bacchelli, 2024).
- Estudos recentes (2023–2025) enfatizam a automação, a Inteligência Artificial (IA) e a documentação mínima orientada ao valor (Tileria; Blasco; Dash, 2024; Naimi *et al.*, 2024; Alzahrani, 2025).

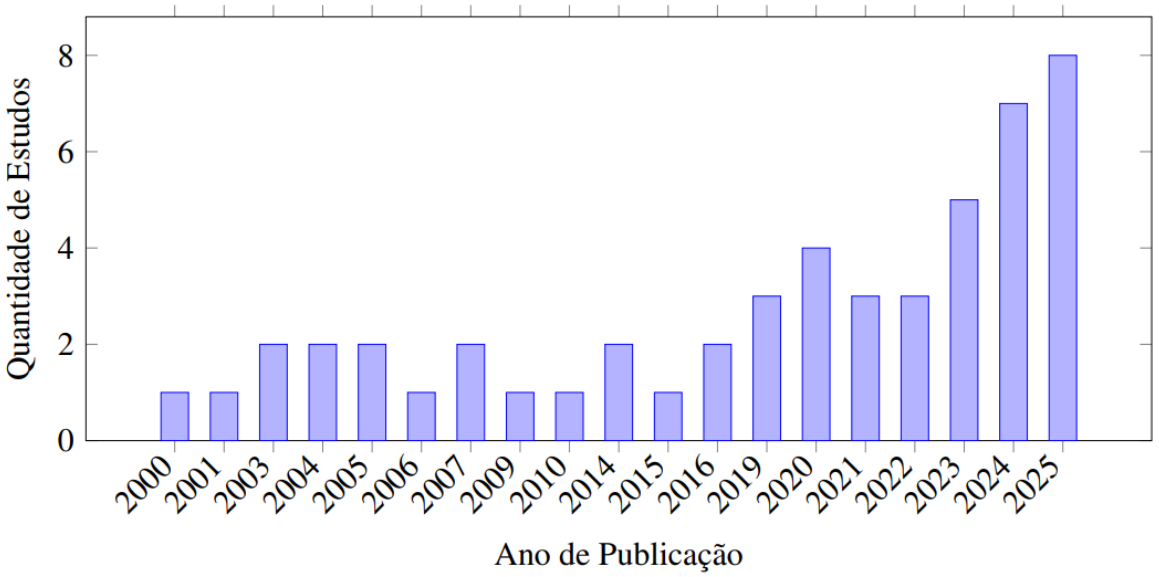
Esse conjunto reafirma que, mesmo em ambientes altamente ágeis e automatizados, a documentação não perdeu a relevância; pelo contrário, tornou-se mais estratégica.

A Figura 1 apresenta a distribuição temporal dos estudos selecionados nesta revisão, evidenciando a evolução das publicações ao longo dos anos. Observa-se um crescimento progressivo a partir de 2015, com uma intensificação significativa entre 2020 e 2025, período que concentra a maior parte dos trabalhos analisados.

Esse aumento reflete o amadurecimento das práticas ágeis, a consolidação do desenvolvimento contínuo (CI/CD) e o surgimento de novas abordagens voltadas à automação e

à documentação viva, indicando que a sobrecarga documental passou a ser tratada como um problema relevante e contemporâneo na Engenharia de Software.

Figura 1 – Distribuição temporal dos estudos selecionados na revisão



Fonte: Autoria própria (2025).

4.2 Principais problemas documentais identificados

Com base nos estudos empíricos analisados, os problemas mais recorrentes foram: desatualização e inconsistências (Romeo *et al.*, 2024; Behutiye *et al.*, 2020); baixa qualidade textual e legibilidade (Bernardo, 2022); excesso de burocracia e artefatos desnecessários (GAO, 2014; Sommerville, 2019); falta de padronização e modelos formais (Santos; Correia, 2020; Monfardini, 2025); resistência do desenvolvedor à documentação (Venigalla; Chimalakonda, 2021); e ausência de responsabilização pelo documento (Coelho, 2009; Brandão, 2019).

Percebe-se que grande parte dos entraves é de origem humana e organizacional, não necessariamente técnica.

4.3 Benefícios da documentação bem estruturada

Apesar dos problemas, os estudos ressaltam que os benefícios superam largamente os custos quando a documentação é (i) mínima e orientada ao valor (Lean e Ágil), (ii) automatizada e integrada ao processo, e (iii) priorizada para quem realmente a utilizará.

Entre os benefícios evidenciados, estão: a redução de erros e retrabalho (Sandhof; Filgueiras, 2006); a melhoria no *onboarding* e a colaboração (Nurlanuly; Araz; Zhuldiz, 2025); a

sustentação da evolução do software (Souza *et al.*, 2004a); e o aumento do desempenho organizacional (Nabot; Al-Qerem, 2025).

Dessa forma, percebe-se que documentar bem implica ganhos de qualidade, comunicação e continuidade.

4.4 Documentação em ambientes ágeis e DevOps

Os estudos convergem para a defesa de uma documentação leve, contínua e integrada ao desenvolvimento, alinhada aos princípios: "*Just Enough Documentation*" (Patton, 2014); "Lean Documentation" (Poppendieck; Poppendieck, 2003); e "*DevOps/Continuous Documentation*" (Theunissen; Heesch; Avgeriou, 2022).

Os times ágeis adotam estratégias como: documentação em código; artefatos gerados automaticamente; validações contínuas; e maior foco em requisitos não funcionais (Behutiye *et al.*, 2022; Nasir *et al.*, 2023). Os resultados sugerem que a dualidade entre documentar e ser ágil não deve existir, já que documentar pode ser um facilitador da agilidade.

Cabe destacar que, embora frequentemente mencionadas em conjunto na literatura, as abordagens de *Lean Documentation* e *Living* ou *Continuous Documentation* possuem naturezas conceitualmente distintas e atuam em níveis complementares do processo de desenvolvimento.

A *Lean Documentation*, inspirada nos princípios do pensamento enxuto, caracteriza-se como uma abordagem predominantemente gerencial, cujo foco está na definição do *que* deve ser documentado. Seu objetivo central é reduzir desperdícios, eliminando artefatos documentais de baixo valor e priorizando informações essenciais para comunicação, manutenção e tomada de decisão (Poppendieck; Poppendieck, 2003; Patton, 2014).

Por outro lado, a *Living* ou *Continuous Documentation* configura-se como uma prática de engenharia de software, relacionada a como a documentação é mantida ao longo do tempo. Essa abordagem enfatiza a integração técnica da documentação aos fluxos de desenvolvimento contínuo, empregando automação, pipelines de CI/CD e sincronização constante entre código e artefatos documentais, conforme discutido por Theunissen, Heesch e Avgeriou (2022).

Dessa forma, enquanto a *Lean Documentation* orienta a redução consciente do escopo documental, a *Continuous Documentation* fornece os mecanismos técnicos necessários para garantir que os artefatos selecionados permaneçam atualizados, coerentes e alinhados à evolução do software. Ambas as abordagens são, portanto, complementares e não equivalentes.

4.5 Contribuições da automação e novas tecnologias

A automação da documentação de software emerge, nos estudos analisados, como uma das tendências mais promissoras para mitigar a sobrecarga documental sem comprometer a qualidade, a rastreabilidade e a confiabilidade das informações técnicas. Diferentemente das abordagens tradicionais, que dependem fortemente de esforço manual e de atualizações periódicas, as soluções automatizadas buscam integrar a geração e a manutenção da documentação diretamente ao fluxo de desenvolvimento.

Os resultados da revisão indicam que as ferramentas de automação documental atuam principalmente em três frentes complementares: (i) extração automática de informações técnicas,

(ii) sincronização contínua entre código e documentação e (iii) geração assistida por inteligência artificial.

No que se refere à extração automática de informações, Tileria, Blasco e Dash (2024) apresentam o *DocFlow*, uma abordagem capaz de extrair especificações técnicas diretamente da documentação existente e do código-fonte, reduzindo a necessidade de intervenção manual. Essa estratégia busca combater o fenômeno do *documentation drift*, no qual a documentação se torna rapidamente obsoleta em relação à implementação, problema amplamente identificado por Romeo *et al.* (2024). Ao automatizar a coleta e a atualização das informações, essas ferramentas contribuem para a redução de inconsistências e retrabalho.

Outra vertente relevante identificada na literatura é a automação integrada a ambientes de desenvolvimento contínuo. Estudos como o de Theunissen, Heesch e Avgeriou (2022) evidenciam que, em contextos de *Continuous Software Development*, a documentação passa a ser tratada como um artefato vivo, atualizado automaticamente a cada ciclo de integração contínua (*CI/CD*). Essa abordagem reforça o conceito de *living documentation*, permitindo que a documentação evolua em sincronia com o software, sem impor custos adicionais significativos às equipes de desenvolvimento.

Mais recentemente, observa-se o crescimento do uso de modelos de linguagem de grande escala (*Large Language Models* – LLMs) como suporte à automação documental. O estudo de Naimi *et al.* (2024) demonstra o potencial dessas tecnologias na geração automática de descrições de casos de uso, na sumarização de requisitos e no apoio à escrita técnica. Diferentemente das abordagens baseadas exclusivamente em regras, os LLMs possibilitam maior flexibilidade semântica, adaptando o conteúdo documental ao contexto do sistema e ao público-alvo. Entretanto, os autores destacam que tais soluções devem ser empregadas como ferramentas de apoio, e não como substitutas completas da validação humana, especialmente em sistemas críticos.

Os resultados também indicam que a automação documental não elimina a necessidade de curadoria e governança. Alzahrani (2025) propõe modelos estruturados, como o *Pre-During-After Software Development Documentation* (*PDA-SDD*), nos quais a automação é distribuída ao longo das fases pré, durante e pós-desenvolvimento. Essa organização evita a geração excessiva de artefatos irrelevantes e reforça o alinhamento entre documentação, processo e objetivos do projeto.

De forma geral, os estudos analisados convergem ao indicar que a automação, quando aplicada de maneira estratégica e contextualizada, contribui significativamente para a redução da sobrecarga documental. Contudo, os resultados também alertam que a adoção indiscriminada de ferramentas automatizadas pode introduzir novos desafios, como documentação redundante, baixa legibilidade ou dependência excessiva de soluções automatizadas. Assim, a automação deve ser compreendida como um meio para potencializar a documentação orientada a valor, e não como um fim em si mesma.

4.6 *Framework* conceitual

A partir da síntese dos estudos analisados na RSL, foi possível estruturar um *framework* conceitual para organizar os principais fatores associados à mitigação da sobrecarga documental em projetos de software. Esse *framework* emerge da convergência dos resultados empíricos, mapeamentos sistemáticos e propostas teóricas identificadas, consolidando-se em quatro pilares fundamentais que orientam práticas documentais mais eficientes, equilibradas e alinhadas aos princípios ágeis e de qualidade.

O objetivo do *framework* não é prescrever um modelo rígido, mas oferecer uma visão estruturada que auxilie equipes de desenvolvimento, gestores e pesquisadores a compreender como a documentação pode agregar valor sem comprometer a agilidade, a produtividade e a qualidade do software.

Figura 2 – *Framework* conceitual proposto para equilibrar a documentação, a agilidade e a qualidade em projetos de software

Fonte: Autoria própria (2025).

4.7 Conclusão dos resultados

A automação reduz custos e mitiga problemas humanos recorrentes, principalmente a desatualização. Entretanto, alerta-se que os modelos precisam de curadoria e validação técnica, o que não substitui a responsabilidade do engenheiro. Os estudos revelam uma forte correlação positiva entre a documentação bem estruturada e a qualidade do produto final, as entregas contínuas e seguras e a capacidade de manutenção a longo prazo.

O enquadramento da documentação como ativo de qualidade é reforçado pela premissa de que uma boa documentação é um investimento (Zhi *et al.*, 2015); melhora o processo de requisitos (Khalid *et al.*, 2025); impacta diretamente no sucesso organizacional (Nabot; Al-Qerem, 2025). Assim, documentar bem é critério de maturidade de software. A documentação de software permanece imprescindível, mas precisa evoluir: deve ser enxuta, automatizada, útil e escrita por quem conhece o produto e seus usuários. O foco deixa de ser "produzir artefatos" e passa a ser gerar conhecimento sustentável.

5 CONCLUSÕES E TRABALHOS FUTUROS

A presente RSL analisou 51 estudos publicados entre 2000 e 2025, com o propósito de investigar a documentação de software sob diferentes perspectivas, em especial no que se refere à sobrecarga documental, às práticas de documentação em ambientes ágeis, à qualidade dos artefatos gerados e ao uso crescente da automação para a mitigação de problemas recorrentes. Os resultados mostram que, apesar da evolução contínua das metodologias de desenvolvimento, a documentação permanece como um elemento essencial para a comunicação entre *stakeholders*, a sustentação da manutenção, a continuidade do projeto e a garantia da qualidade do produto de software.

O estudo confirmou que a problemática da sobrecarga documental emerge principalmente de um desalinhamento entre os artefatos produzidos e suas reais necessidades de uso. Modelos tradicionais baseados em extensos registros textuais geram desperdícios, atrasos, retrabalho e desmotivação das equipes, especialmente em contextos de desenvolvimento ágil, nos quais o tempo é um fator crítico. Entretanto, longe de representar uma contradição, o desenvolvimento ágil reforça a importância de uma documentação suficiente, objetiva e fluida, centrada no valor e não na burocracia.

Outro achado relevante foi que grande parte dos problemas associados à documentação não é de ordem técnica, mas humana e organizacional, incluindo baixa priorização, ausência de padrões, falta de revisões contínuas, desconhecimento das melhores práticas e resistência cultural. Tais fatores indicam que a documentação eficaz é resultado direto de governança, políticas internas e formação profissional adequada, mais do que de ferramentas ou métodos isolados.

Os estudos mais recentes mostram que tecnologias como automação documental, IA, análise estática e extração automatizada de artefatos têm se tornado fortes aliadas na mitigação da defasagem entre código, *design* e documentos. Contudo, mesmo em cenários altamente automatizados, ressalta-se que a validação humana permanece indispensável, pois os modelos de processamento de linguagem natural ainda carecem de precisão contextual e de entendimento semântico pleno.

Com base na convergência das evidências encontradas, conclui-se que uma abordagem documental eficiente deve se apoiar em quatro pilares fundamentais: (i) *Valor*: documentar apenas o que apoia decisões, comunicação ou manutenção futura; (ii) *Atualização contínua*: incorporar a documentação ao fluxo de desenvolvimento; (iii) *Padronização e legibilidade*: documentos pensados para seus usuários reais; e *Automação inteligente*: máquinas como suporte,

pessoas como responsáveis.

Portanto, o desafio contemporâneo não está mais em “documentar mais”, mas sim em documentar melhor. A documentação de software deve ser tratada como um ativo estratégico de qualidade e evolução tecnológica, refletindo um sistema vivo que cresce, adapta-se e se aprimora junto ao produto que descreve.

Com base na literatura analisada, é importante que futuros projetos: adotem políticas organizacionais claras sobre artefatos mínimos e responsáveis; utilizem automação para rastreabilidade e integração contínua; promovam treinamentos sobre escrita técnica e boas práticas documentais; priorizem documentação em código e alinhamento direto com requisitos de qualidade; e ampliem pesquisas com foco em métricas, modelagem semântica e sustentabilidade documental.

Em síntese, a documentação não é um custo, mas um investimento. Seu valor se manifesta no tempo, na qualidade e na capacidade do software de permanecer funcional, compreensível e evolutivo. Uma documentação eficaz transforma conhecimento tácito em patrimônio orga- nizacional e reduz o impacto da rotatividade, da complexidade e da passagem do tempo. O futuro do desenvolvimento de software depende, em grande parte, da capacidade de preservar, compartilhar e aprimorar o conhecimento produzido.

REFERÊNCIAS

ALZAHRANI, A. A. H. Pre-during-after software development documentation (pda-sdd): A phase-based approach for comprehensive software documentation in modern development paradigms. **Computers**, v. 14, n. 9, p. 378, 2025. https://www.mdpi.com/2073-431X/14/9/378.

ARYA, D. M.; GUO, J. L. C.; ROBILLARD, M. P. Why people contribute software documentation. In: **Proceedings of the 2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering**. [S.l.: s.n.], 2024. p. 91–96. https://dl.acm.org/doi/10.1145/3641822.3641881.

BECK, K. *et al.* **Manifesto para Desenvolvimento Ágil de Software**. 2001. https://agilealliance. org/agile101/the-agile-manifesto/.

BEHUTIYE, W. *et al.* Towards optimal quality requirement documentation in agile software development: A multiple case study. **Journal of Systems and Software**, v. 183, p. 111112, 2022. https://www.sciencedirect.com/science/article/pii/S0164121221002090.

BEHUTIYE, W. *et al.* Documentation of quality requirements in agile software development. In: **Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2020. (EASE '20), p. 250–259. ISBN 9781450377317. https://dl.acm.org/doi/10.1145/3383219.3383245.

BERNARDO, K. W. N. **Documentação de Software: Exploração dos problemas mais frequentes e qualidades mais relevantes**. 2022. https://repositorio.ufpb.br/jspui/handle/123456789/31628.

BRANDãO, B. V. V. **Documentação de software: uma abordagem possível para micro e pequenas empresas**. 2019. https://repositorio.ueg.br/jspui/handle/riueg/5389.

CARLOS, W. F. d. P. D.; HADDAD, F. B. B. Estruturação da documentação de software em uma equipe de desenvolvimento low-code. In: **Simpósio Brasileiro de Sistemas de Informação (SBSI)**. [S.l.]: SBC, 2025. p. 241–246. https://sol.sbc.org.br/index.php/sbsi_estendido/article/view/34613.

COELHO, H. S. Documentação de software: uma necessidade. **Texto Livre: linguagem e tecnologia**, v. 2, n. 1, p. 17–21, 2009. https://www.redalyc.org/pdf/5771/577163636007.pdf.

DEBASTIANI, C. A. **Definindo escopo em projetos de software**. [S.l.]: Novatec, 2016. https://books.google.com.br/books?hl=pt-BR&lr=&id=w255DAAAQBAJ&oi=fnd&pg=PT4&dq=DEBASTIANI.

FRANCA, A. J. d. *et al.* Linkdoc: An automated process in the delivery of documentation in a global software development environment. In: **Proceedings of the 5th World Symposium on Software Engineering (WSSE 2023)**. [S.l.: s.n.], 2023. p. 9–16. https://dl.acm.org/doi/10.1145/3631991.3632016.

GAO, U. S. G. A. O. **Healthcare.gov: CMS Management of the Federal Marketplace**. [S.l.]:

GAO, 2014. https://apps.dtic.mil/sti/trecms/pdf/AD1176090.pdf.

GAROFALO, S.; ALIS, G. Cerceamento ou acessibilidade: uma discussão sobre a documentação em software livre. **Texto Livre**, v. 3, n. 2, p. 25–32, 2010. https://periodicos.ufmg.br/index.php/ textolivre/article/view/16579.

GOMES, A. **Elicitando requisitos em projetos de Software Educativo**. 2000. https://sl1nk.com/Z9Hu8.

HOY, Z.; XU, M. Agile software requirements engineering challenges-solutions—a conceptual framework from systematic literature review. **Information**, v. 14, n. 6, p. 322, 2023. https://www.mdpi.com/2078-2489/14/6/322.

KHALID, S. *et al.* Ensuring quality in software requirement engineering process: A comparative study. **Egyptian Informatics Journal**, v. 31, 2025. https://www.sciencedirect.com/science/article/pii/S1110866525001471.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. **Information and Software Technology**, 2007. https://www.sciencedirect. com/science/article/pii/S016412120600197X.

KLOCK, A. C. T. Mapeamentos e revisões sistemáticos da literatura: um guia teórico e prático. **Cadernos de Informática**, v. 13, n. 1, 2021. https://seer.ufrgs.br/cadernosdeinformatica/article/ view/118671.

KONNORATE, G. C. *et al.* A importancia do controle de versões no desenvolvimento de software. In: **Seminário de Tecnologia Gestão e Educação**. [S.l.: s.n.], 2019. v. 1, n. 2, p. 1–4. https://raam.alcidesmaya.edu.br/index.php/SGTE/article/view/9.

KUBIACZYK, D.; JARZE˛BOWICZ, A. Non-functional requirements documentation techniques in agile software development: A focus group study. 2025. https://aisel.aisnet.org/isd2014/proceedings2025/agile/13/.

LACERDA, G. S. d. **FrameworkDoc: ferramenta de documentação e geração de artefatos de software**. 2005. https://lume.ufrgs.br/handle/10183/6027.

LEHNER, F. Quality control in software documentation: Measurement of text comprehensibility. **Information & Management**, v. 25, n. 3, p. 133–146, 1993. https://www.sciencedirect.com/ science/article/pii/037872069390036S.

LENT, B. **Critical review of agile approach in project management: phenomenological exploration**. 2025. https://www.researchgate.net/publication/395335691.

MARQUES, V. S. S. **Engenharia de software aplicada a softwares biológicos: um estudo de caso em documentação de software bioinformático**. 2021. https://repositorio.ufu.br/handle/123456789/34046.

MARTINS, J. C. C. **Técnicas para gerenciamento de projetos de software**. [S.l.]: Brasport, 2007. https://books.google.com.br/books?hl=pt-BR&lr=&id=Axl2RZQdE68C.

MENDES, R. B. d. A. **Gestão e processamento de erros e configuração de software**. Tese (Doutorado), 2007. https://www.proquest.com/openview/7a3a7f1ef2e48b9f31f8f1eb193fd13e/1?pq-origsite=gscholar&cbl=2026366&diss=y.

MONFARDINI, N. O papel do redator-revisor técnico na documentação de software: tradução técnica, manualização e revisão de textos. **Revista de Produção Editorial**, v. 5, 2025. https://periodicos.ufsm.br/gutenberg/article/view/85503/62855.

NABOT, A.; AL-QEREM, A. Impact of software quality on organizational performance. **Array**,p. 100476, 2025. https://www.sciencedirect.com/science/article/pii/S2590005625001031.

NAIMI, L. *et al.* Automating software documentation: Employing llms for precise use case description. **Procedia Computer Science**, v. 246, p. 1346–1354, 2024. https://www.sciencedirect. com/science/article/pii/S1877050924026176.

NASIR, S. *et al.* An exploratory study about non-functional requirements documentation prac- tices in agile teams. In: **Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2023. (SAC '23), p. 1009–1017. ISBN 9781450395175. https://dl.acm.org/doi/10.1145/3555776.3577605.

NASSIF, M.; ROBILLARD, M. P. Non-linear software documentation with interactive code examples. **ACM Trans. Softw. Eng. Methodol.**, Association for Computing Machinery, New York, NY, USA, v. 34, n. 2, jan. 2025. ISSN 1049-331X. https://dl.acm.org/doi/10.1145/3702976. Disponível em: https://doi.org/10.1145/3702976.

NURLANULY, M. A.; ARAZ, Z. A.; ZHULDIZ, A. Analysis the impact of code documentation styles on collaborative software development. **Universum: Tekhnicheskie Nauki**, v. 7, n. 5(134), p. 48–55, 2025. Disponível em: https://cyberleninka.ru/article/n/analysis-the-impact-of-code-documentation- styles-on-collaborative-software-development.

PATTON, J. **User Story Mapping**. [S.l.]: O'Reilly, 2014. https://books.google.com.br/books?id=4YZyBAAAQBAJ.

PINHO, F. G. d. N. **Desafios e soluções na documentação em projetos de software livre e de código aberto: um mapeamento sistemático da literatura**. 2024. https://repositorio.ufc.br/handle/riufc/77431.

POPPENDIECK, M.; POPPENDIECK, T. **Lean Software Development: An Agile Toolkit**. [S.l.]: Addison-Wesley, 2003. https://books.google.com.br/books?id=IJ1gAgAAQBAJ.

REIS, C. R. **Caracterização de um processo de software para projetos de software livre**. Dissertação (Mestrado) — ICMC, USP, São Carlos, Brasil, 2003.

ROMEO, J. *et al.* Capturing and understanding the drift between design, implementation, and documentation. In: **Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension**. New York, NY, USA: Association for Computing Machinery, 2024. (ICPC '24), p. 382–386. ISBN 9798400705861. https://dl.acm.org/doi/10.1145/3643916.3644399. Disponível em:

https://doi.org/10.1145/3643916.3644399.

SANDHOF, K.; FILGUEIRAS, L. Defeitos de software como erros humanos. In: **II Workshop Um olhar sociotécnico sobre a Engenharia de Software (WOSES 2006)**. [S.l.: s.n.], 2006. https://www.researchgate.net/profile/Lucia-Filgueiras/publication/266356181_Defeitos_de_Software_como_Erros_Humanos/links/5519338d0cf2d241f355c1f0/Defeitos-de-Software-como-Erros-Humanos.pdf.

SANTOS, A. F. d. **Documentação de Software para usuários desenvolvedores: uma avaliação de bibliotecas ORM populares na plataforma Node**. Trabalho de Conclusão de Curso — Universidade Federal do Rio Grande do Norte, 2025. https://repositorio.ufrn.br/server/api/core/ bitstreams/74c6275d-fab2-45b6-bb78-77eebfe26417/content.

SANTOS, G. S. D. **Um processo de representação estruturada e validação de requisitos de software para mitigar problemas semânticos**. Tese (Doutorado), 2021. https://hdl.handle.net/20.500.12733/2743.

SANTOS, J.; CORREIa, F. F. A review of pattern languages for software documentation. In: **Proceedings of the European Conference on Pattern Languages of Programs 2020**. [S.l.: s.n.], 2020. p. 1–14. https://dl.acm.org/doi/10.1145/3424771.3424786.

SILVA, A. S. d. F. **Documentação de software: uma análise comparativa entre documentação tradicional e living documentation**. Dissertação de Mestrado — Universidade Federal do Rio Grande do Norte, 2020. https://repositorio.ufrn.br/items/3794631e-10fa-4e5a-a55c-4d2c80d94a1c.

SILVA, E. C. D.; LOVATO, L. A. Framework scrum: eficiência em projetos de software. **Gestão e Projetos: GeP**, v. 7, n. 2, p. 1–15, 2016. https://dialnet.unirioja.es/servlet/articulo?codigo=5632152.

SOMMERVILLE, I. **Engenharia de Software**. 10. ed. [S.l.]: Pearson, 2019. https://archive.org/details/sommerville-engenharia-de-software-10e.

SOUZA, S. C. B. de *et al.* Documentação essencial para manutenção de software ii. In: **Workshop de Manutenção de Software Moderna**. Brasília, Brasil: [s.n.], 2004a. https://www.academia.edu/download/39852944/Documentao_Essencial_para_Manuteno_de_So20151109-23263-1ua4c6j.pdf.

SOUZA, S. C. B. de *et al.* Investigação da documentação de maior importância para manutenção de software. In: **Jornada Iberoamericanas en Ingeniería del Software e Ingeniería del Co- nocimiento**. [S.l.: s.n.], 2004b. p. 1–14. http://www.grise.upm.es/rearviewmirror/conferencias/ jiisic04/Papers/49.pdf.

SOVRANO, F.; LOGNOUL, M.; BACCHELLI, A. An empirical study on compliance with ranking transparency in the software documentation of EU online platforms. In: **Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society**. [S.l.: s.n.], 2024. p. 46–56. https://dl.acm.org/doi/10.1145/3639475.3640112.

THEUNISSEN, T.; HEESCH, U. V.; AVGERIOU, P. A mapping study on documentation in

continuous software development. **Information and Software Technology**, v. 142, p. 106733, 2022. https://www.sciencedirect.com/science/article/pii/S095058492100183X.

TILERIA, M.; BLASCO, J.; DASH, S. K. Docflow: Extracting taint specifications from software documentation. In: **Proceedings of the 46th IEEE/ACM International Conference on Software Engineering**. [S.l.: s.n.], 2024. p. 1–12. https://dl.acm.org/doi/10.1145/3597503.3623312.

VENIGALLA, A. S. M.; CHIMALAKONDA, S. Understanding emotions of developer community towards software documentation. In: **2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)**. [S.l.]: IEEE, 2021. p. 87–91. https://dl.acm.org/doi/10.1109/ICSE-SEIS52602.2021.00018.

VENIGALLA, A. S. M.; CHIMALAKONDA, S. An exploratory study of software artifacts on GitHub from the lens of documentation. **Information and Software Technology**, v. 169, p. 107425, 2024. https://www.sciencedirect.com/science/article/pii/S0950584924000302?casa_token=Rxyj9uIjqTYAAAAA:GoT0AOZF7hcAci2lLwVnTTBoe-ugRtVQmo5rdEU4EdLA8T-Und43iIuzcrybP9rNzb8jikWsHQ.

YAMANOUCHI, J. R. **A importância da documentação no processo de desenvolvimento de sistemas**. 2020. http://ric.cps.sp.gov.br/handle/123456789/4811.

ZANG, K. R.; DENARDI, M. A. **Documentação de software para aplicações web**. 2005. https://repositorio.pgsscogna.com.br/bitstream/123456789/12642/1/DOCUMENTA%C3%87%C3%83O%20DE%20SOFTWARE%20PARA%20APLICA%C3%87%C3%95ES%20WEB.pdf.

ZHI, J. *et al.* Cost, benefits and quality of software development documentation: A systematic mapping. **Journal of Systems and Software**, v. 99, p. 175–198, 2015. https://www.sciencedirect. com/science/article/abs/pii/S0164121214002131.

APÊNDICE A – CLASSIFICAÇÃO E EXTRAÇÃO DOS ESTUDOS SELECIONADOS

Tabela 2 – Classificação e extração dos estudos selecionados

ID	Referência	Ano	Tipo de Estudo	Contribuição Principal	Qualidade
E01	Beck *et al.* (2001).	2001	Manifesto	Prioriza software funcional em vez de documentação extensa.	Alta
E02	Hoy e Xu (2023)	2023	Revisão Sistemática	Desafios e soluções para requisitos em ambientes ágeis.	Alta
E03	Lent (2025)	2023	Estudo crítico	Limitações do ágil em gestão de grandes projetos.	Média
E04	Patton (2014)	2014	Livro	*User Story Mapping* e documentação suficiente.	Alta
E05	Poppendieck e Poppendieck (2003)	2003	Livro	Lean Documentation reduz desperdícios documentais.	Alta
E06	Sommerville (2019)	2019	Livro	Riscos da superdocumentação e requisitos mutáveis.	Alta
E07	GAO (2014)	2014	Relatório técnico	Falhas documentais em healthcare.gov devido à burocracia.	Alta
E08	Coelho (2009)	2009	Artigo	Documentação como elemento essencial para comunicação.	Média
E09	Silva (2020)	2020	Dissertação	Comparativo entre documentação tradicional e viva.	Alta
E10	Pinho (2024)	2024	Mapeamento Sistemático	Desafios em documentação FLOSS.	Alta
E11	Bernardo (2022)	2022	Estudo empírico	Problemas e qualidades mais relevantes na documentação.	Média/Alta
E12	Garofalo e Alis (2010)	2010	Artigo	Documentação e acessibilidade no software livre.	Média
E13	Carlos e Haddad (2025)	2025	Estudo de caso	Documentação em equipes *low-code*.	Média/Alta
E14	Debastiani (2016)	2016	Livro	Documentação aplicada ao escopo do projeto.	Alta
E15	Reis (2003)	2003	Dissertação	Mínimo documental em FLOSS.	Média
E16	Gomes (2000)	2000	Dissertação	Elicitação em software educacional apoiada por documentação.	Média

Continua na próxima página

Continuação da Tabela 2

ID	Referência	Ano	Tipo de Estudo	Contribuição Principal	Qualidade
E17	Martins (2007)	2007	Livro	Documentação no gerenciamento de projetos.	Alta
E18	Silva e Lovato (2016)	2016	Artigo	Eficiência da documentação mínima no Scrum.	Média/Alta
E19	Arya, Guo e Robillard (2024).	2024	Estudo empírico	Motivações para contribuição em documentação.	Alta
E20	Sovrano, Lognoul e Bacchelli (2024).	2024	Estudo empírico	Transparência documental em plataformas digitais.	Alta
E21	Tileria, Blasco e Dash (2024)	2024	Artigo técnico	DocFlow: automação da extração documental.	Alta
E22	Franca *et al.* (2023)	2023	Artigo técnico	Automação documental distribuída (GSD).	Média/Alta
E23	Santos e Correia (2020)	2020	Revisão	Padrões documentais em software.	Média
E24	Romeo *et al.* (2024)	2024	Estudo empírico	Monitoramento do desvio entre código e documentação.	Alta
E25	Venigalla e Chimalakonda (2021)	2021	Estudo empírico	Emoções negativas ligadas à documentação ruim.	Média/Alta
E26	**??**)	2025	Estudo de campo	Formatos documentais preferidos por desenvolvedores.	Alta
E27	Behutiye *et al.* (2020)	2020	Estudo empírico	Requisitos de qualidade em equipes ágeis.	Alta
E28	Nasir *et al.* (2023)	2023	Exploratório	Documentação de requisitos não-funcionais em ágil.	Média/Alta
E29	Yamanouchi (2020)	2020	Relato técnico	Documentação como suporte à continuidade do sistema.	Média
E30	Monfardini (2025)	2025	Artigo	Revisão técnica e padronização textual.	Média/Alta
E31	Konnorate *et al.* (2019)	2019	Artigo	Controle de versões como elemento documental.	Média

Continua na próxima página

Continuação da Tabela 2

ID	Referência	Ano	Tipo de Estudo	Contribuição Principal	Qualidade
E32	Souza *et al.* (2004b)	2004	Estudo empírico	Documentos prioritários na manutenção.	Alta
E33	Zang e Denardi (2005)	2005	Artigo	Documentação necessária em aplicações web.	Média
E34	Sandhof e Filgueiras (2006)	2006	Artigo	Erros humanos derivados de documentação inadequada.	Média/Alta
E35	Santos (2025)	2025	TCC	Bibliotecas ORM: documentação para devs.	Média
E36	Mendes (2007)	2007	Tese	Documentação de configuração e erros.	Média/Alta
E37	Santos (2021)	2021	Tese	Validação e semântica de requisitos documentados.	Alta
E38	Brandão (2019)	2019	Artigo	Documentação mínima para pequenas empresas.	Média
E39	Marques (2021)	2021	Estudo de caso	Documentação em sistemas bioinformáticos.	Média/Alta
E40	Lacerda (2005)	2005	Artigo técnico	FrameworkDoc para automação documental.	Média/Alta
E41	Venigalla e Chimalakonda (2021)	2024	Estudo empírico	Artefatos no GitHub sob análise documental.	Alta
E42	Theunissen, Heesch e Avgeriou (2022)	2022	Mapeamento Sistemático	Documentação em CI/CD contínuo.	Alta
E43	Nabot e Al-Qerem (2025)	2025	Estudo empírico	Qualidade documental impacta desempenho organizacional.	Alta
E44	Khalid *et al.* (2025)	2025	Comparativo	Qualidade documental na engenharia de requisitos.	Alta
E45	Behutiye *et al.* (2022)	2022	Casos Múltiplos	Documentação otimizada de NFRs.	Alta
E46	Zhi *et al.* (2015)	2015	Mapeamento Sistemático	Custos e benefícios da documentação.	Alta
E47	Naimi *et al.* (2024)	2024	Artigo técnico	LLMs para automatizar casos de uso.	Alta

Continua na próxima página

Continuação da Tabela 2

ID	Referência	Ano	Tipo de Estudo	Contribuição Principal	Qualidade
E48	Nurlanuly, Araz e Zhuldiz (2025)	2025	Estudo empírico	Estilos de documentação e colaboração.	Alta
E49	Alzahrani (2025)	2025	Framework	PDA-SDD para documentação completa.	Alta
E50	Kubiaczyk e Jarze¸bowicz (2025)	2025	Qualitativo	Técnicas para documentar NFRs em ágil.	Alta
E51	Souza *et al.* (2004a)	2007	Artigo	Documentos essenciais na manutenção.	Alta

Fonte: Autoria própria (2025).