Research, Society and Development, v. 15, n. 4, e8615450966, 2026
(CC BY 4.0) | ISSN 2525-3409 | DOI: http://dx.doi.org/10.33448/rsd-v15i4.50966

O uso do assistente de programação baseado em Inteligência Artificial, o GitHub Copilot, na Qualidade de Software: Uma revisão sistemática de literatura

The use of the AI-based programming assistant, GitHub Copilot, in Software Quality: A systematic literature review

El uso del asistente de programación basado en IA, GitHub Copilot, en la Calidad del Software: Una revisión sistemática de la literatura

Recebido: 14/04/2026 | Aceito: 23/04/2026 | Publicado: 24/04/2026

Laís Aparecida Ferreira de Oliveira
ORCID: https://orcid.org/0009-0009-5847-8895
Faculdade de Tecnologia da Zona Leste, Brasil
E-mail: laisestudante16@gmail.com
Cristina Corrêa de Oliveira
ORCID: https://orcid.org/0000-0002-8629-6679
Faculdade de Tecnologia da Zona Leste, Brasil
E-mail: cristina.oliveira@fatec.sp.gov.br

Resumo
O objetivo do artigo é investigar como a literatura científica tem relatado o uso do GitHub Copilot e seu impacto na qualidade de código. A pesquisa foi realizada por meio de uma revisão sistemática da literatura, seguindo etapas de definição dos critérios de inclusão e exclusão, busca em bases de dados disponibilizadas pelo Portal de Periódicos CAPES, seleção dos estudos relevantes e análise qualitativa dos selecionados. Foram identificados e analisados quinze estudos que abordam diferentes aspectos do GitHub Copilot, incluindo o aspecto de segurança, a influência da estrutura dos *prompts*, os impactos na produtividade, a comparação sobre desempenho em relação aos humanos e os desafios enfrentados pelos desenvolvedores, evidenciando tanto benefícios quanto limitações no uso da ferramenta. A ferramenta apresenta-se como um apoio útil ao desenvolvedor, servindo como um bom ponto de partida e direcionamento para a codificação. Entretanto, devido a potenciais tendências de geração de código inseguro e dificuldade em lidar com tarefas complexas, não é recomendado a ser usado de modo isolado. Além disso, os resultados também dependem da forma como os desenvolvedores utilizam o GitHub Copilot, sendo que a confiança excessiva e *prompts* mal estruturados podem levar a retrabalho, especialmente entre programadores iniciantes, público predominante da ferramenta.
Palavras-chave: Copilot; Qualidade de Software; Desenvolvimento de Software; Automação no Desenvolvimento.

Abstract
The purpose of this paper is to investigate how the scientific literature has reported the use of GitHub Copilot and its impact on code quality. The research was conducted through a systematic literature review, following steps such as defining inclusion and exclusion criteria, searching in databases available through the CAPES Portal of Journals, selecting relevant studies, and performing a qualitative analysis of the selected works. Fifteen studies were identified and analyzed, addressing different aspects of GitHub Copilot, including security issues, the influence of prompt structure, impacts on productivity, performance comparisons with human developers, and the challenges faced by users. The findings highlight both the benefits and limitations of using the tool. The tool proves to be a useful support for developers, serving as a good starting point and guide for coding tasks. However, due to its tendency to generate insecure code and its limited capacity to handle complex tasks, it is not recommended for use in isolation. Moreover, outcomes also depend on how developers interact with GitHub Copilot, as excessive reliance and poorly structured prompts may lead to rework — especially among novice programmers, who represent the tool's primary user base.
Keywords: Copilot; Software Quality; Software Development; Development Automation.

Resumen
El objetivo de este artículo es investigar cómo la literatura científica ha reportado el uso de GitHub Copilot y su impacto en la calidad del código. La investigación se llevó a cabo mediante una revisión sistemática de la literatura, siguiendo etapas como la definición de los criterios de inclusión y exclusión, la búsqueda en bases de datos disponibles a través del Portal de Periódicos CAPES, la selección de estudios relevantes y el análisis cualitativo de los trabajos seleccionados. Se identificaron y analizaron quince estudios que abordan diferentes aspectos de GitHub Copilot, incluyendo cuestiones de seguridad, la influencia de la estructura de los prompts, los impactos en la productividad,

comparaciones de rendimiento con desarrolladores humanos y los desafíos enfrentados por los usuarios. Los hallazgos destacan tanto los beneficios como las limitaciones del uso de la herramienta. La herramienta se presenta como un apoyo útil para los desarrolladores, sirviendo como un buen punto de partida y guía para las tareas de codificación. Sin embargo, debido a su tendencia a generar código inseguro y su limitada capacidad para manejar tareas complejas, no se recomienda su uso de forma aislada. Además, los resultados también dependen de cómo los desarrolladores interactúan con GitHub Copilot, ya que la confianza excesiva y los prompts mal estructurados pueden llevar a retrabajo, especialmente entre los programadores principiantes, quienes representan la principal base de usuarios de la herramienta.
Palabras clave: Copilot; Calidad del Software; Desarrollo de Software; Automatización en el Desarrollo.

1. Introdução

A cada ano que passa, a adoção da Inteligência Artificial (IA) nas organizações tem se expandido e consolidado de maneira acelerada. Isso é sinalizado na pesquisa global da International Data Corporation (IDC) — empresa de consultoria e pesquisa global de tecnologia da informação (About IDC, 2024) — em parceria com a Microsoft, que apresentou um estudo que o uso da IA generativa nas empresas saltou de 55% em 2023 para 75% em 2024 (Jyoti & Schubmehl, 2024, p. 6).

Essa tendência se reflete no segmento de desenvolvimento de software, onde o interesse do uso da IA por parte dos programadores também cresce. Um levantamento realizado pelo Stack Overflow em maio de 2024 com mais de 65 mil desenvolvedores ao redor do mundo mostrou que 76% deles já utilizam ou pretendem utilizar ferramentas de IA, o que representa um aumento de 6% em relação ao ano anterior (Stack Overflow, 2024).

Um dos marcos significativos da popularização da IA generativa deu-se com o lançamento do ChatGPT em 2022, que conquistou 100 milhões de usuários e 13 milhões de visitantes diários no site em apenas dois meses (McKinsey & Company, 2023). O fator chamativo para sua fama se deu pela forma de interatividade que os usuários poderiam utilizar, que são os diálogos com a IA. Em relação aos programadores, estes poderiam buscar suporte para manutenção e sugestões de código por meio de interação com escrita informal (OpenAI, 2022), diferenciando-se do modo tradicional de busca online em fóruns e outros materiais.

O fato é que a proeminência do ChatGPT influenciou outras empresas, líderes em tecnologia, a desenvolverem ferramentas similares — como o Gemini, do Google — e impulsionasse a popularidade de outras já existentes, como o GitHub Copilot, lançado pela primeira vez em outubro de 2021, presente em mais de 4 milhões de organizações ao redor do mundo (GitHub, s.d.a).

O objetivo geral do artigo é investigar como a literatura científica tem relatado o uso do GitHub Copilot e seu impacto na qualidade de código. Por ser uma fase essencial no processo de desenvolvimento de software, a codificação influencia diretamente características como manutenibilidade e confiabilidade do produto final. Assim, para alcançar esse objetivo geral, foram definidos os seguintes objetivos específicos que nortearam a pesquisa:

1) Investigar como a literatura científica tem avaliado a qualidade de código gerado com o uso do GitHub Copilot: Verificar o que as pesquisas empíricas e experimentais dessa tecnologia trazem acerca de seus benefícios e limitações quanto à sua aplicação na codificação.

2) Analisar as percepções dos desenvolvedores quanto ao impacto da ferramenta em relação à produtividade e ao processo de codificação: Realizar um levantamento de estudos anteriores nos quais foram aplicados questionários e experimentos com profissionais da área sobre o uso do GitHub Copilot, a fim de identificar os benefícios e limitações percebidos em seu fluxo de trabalho.

Para seleção dos estudos que atendam esses objetivos, foi adotado o processo de pesquisa científica *Knowledge Development Process - Constructivist* (ProKnow-C), cuja finalidade visa auxiliar pesquisadores na seleção de artigos relevantes com uso de técnicas de análise bibliométrica.

2. Fundamentação Teórica

A seguir, a fim de embasar esta pesquisa, esta seção apresenta um breve contexto sobre a produção de software, destacando o impacto da dívida técnica que surge, eventualmente, no desenvolvimento e afeta o futuro do produto. Em seguida, é abordada a capacidade da inteligência artificial como uma ferramenta potencial para mitigar esses desafios. Por fim, são exploradas as características do GitHub Copilot, com ênfase na compreensão de sua natureza, capacidades atuais e de como a ferramenta gera sugestões de código durante o processo de desenvolvimento.

2.1 Os desafios da produção de um software de qualidade

A incorporação do software na vida pessoal e nas organizações é reflexo da transformação digital que caracteriza o século XXI e atualmente é uma das tecnologias mais importantes em escala global. Isso justifica a importância da procura de alternativas de aperfeiçoamento de processos para criação de novos produtos de software de qualidade, uma vez que ainda há desafios em relação a entregas dentro do prazo e orçamento (Pressman & Maxim, 2021, p. 73). Contudo, sem o devido cuidado com a maneira de desenvolver um software, pode-se acarretar um problema denominado dívida técnica.

Na Engenharia de Software, este conceito diz respeito à consequência de entregas de softwares mais rápidas, mas não sustentáveis, que podem acarretar custos que serão posteriormente revertidos em financiamentos para consertar as escolhas de baixa qualidade feitas em prol de um cumprimento de prazo ou requisito imediato (RocketCode, 2023). Pode acontecer em qualquer fase do desenvolvimento de software, seja na análise, projeto arquitetural ou na etapa de codificação, espalhando-se por todas as partes do sistema (Krasner, 2022, p. 28). A utilização de uma palavra comumente associada a assuntos financeiros remete à ideia de que a falta de gerenciamento para quitar esse débito pode acarretar consequências negativas no futuro do projeto (Paula, 2024).

Conforme Krasner (2022), a estimativa de custo da má qualidade de software cresceu para pelo menos US$ 2,41 trilhões de dólares, com a dívida técnica acumulada a aproximadamente US$ 1,52 trilhões de dólares, porque as falhas nos sistemas não estão sendo corrigidas (Krasner, 2022, p. 4). O autor apresenta que a Dívida Técnica se divide em duas partes (Krasner, 2022, p. 28):

1) Principal: É a primeira parte do débito, refere-se ao custo de modificar os artefatos de software para que elevem o nível de manutenibilidade e evolução.

2) Juros: Refere-se ao esforço extra que os desenvolvedores gastam para realizar as alterações por conta desse débito que se acumula ao longo do tempo.

Embora tenha apresentado as falhas dos softwares e suas razões, Krasner (2022, p. 41) também propõe algumas soluções para reduzir esses danos, e uma delas é o uso da Inteligência Artificial.

2.2 Inteligência Artificial

De acordo com o dicionário Priberam, o adjetivo “revolucionário” é definido como o “que introduz novidades ou grandes alterações” (Priberam, 2024). Nesse sentido, a Inteligência Artificial é uma tecnologia que se destaca por ser revolucionária, pois sua utilização e aplicação gera um impacto e transforma a maneira como as coisas são feitas. Essa tecnologia é portadora de competências que a tornam uma ferramenta de apoio para solucionar dificuldades que um humano não tem alçada para resolver (Sena *et al*., 2024, p. 7) assim como também é capaz de sistematizar e automatizar tarefas, tornando-a efetivamente uma ferramenta colaboradora para as atividades humanas (Sena *et al.*, 2024, p. 3). Isso provém da sua formação de componentes algorítmicos, matemáticos e redes neurais que lhe permitem uma aprendizagem através de análise de dados e reconhecimento de padrões (Sena *et al.*, 2024, p. 6).

2.3 GitHub Copilot e suas características

O GitHub Copilot é uma ferramenta de assistente de codificação, fruto de uma parceria da empresa OpenAI — uma organização estadunidense de pesquisa e desenvolvimento de Inteligência Artificial — e GitHub — uma plataforma de hospedagem e compartilhamento de repositórios de código.

Fazendo jus ao significado literal de seu sobrenome, o GitHub Copilot foi lançado com o objetivo de ser um "copiloto" — um suporte virtual para os desenvolvedores — devido às suas características e habilidades que impactam o processo de programar, proporcionando uma direção mais clara para o desenvolvedor ter um caminho a seguir em seu trabalho. Segundo a documentação oficial, algumas funcionalidades atuais do Copilot incluem: sugestões de códigos enquanto ocorre a digitação no ambiente de desenvolvimento integrado (IDE), interação por meio de chat para tirar dúvidas de programação e auxílio via linha de comando (GitHub, s.d.c).

Essa capacitação vem da sua natureza, composta por duas subcategorias de Inteligência Artificial: *Large Language Model* (LLM) e a IA Generativa.

O LLM é um modelo que, baseado em uma grande quantidade de dados de texto, aprende a encontrar padrões e assim consegue prever quais são as próximas palavras a aparecer após as outras (Lopes, 2023). No caso do modelo Codex, tecnologia base do Copilot, o treinamento vem de uma vasta coleção de textos e códigos-fonte disponíveis em repositórios públicos na plataforma GitHub, segundo informações do site oficial (GitHub, s.d.b).

A IA Generativa é capaz de gerar novo conteúdo a partir do existente (Sauvola *et al.*, 2024, p.3). No contexto de programação, pode oferecer sugestões preditivas de código (Song *et al.*, p. 2).

De forma prática, conforme explicado pelo *Frequently Asked Questions* (FAQ) do próprio site oficial do GitHub Copilot (GitHub, 2025), a ferramenta traz a recomendação de código seguindo esses passos:

Primeiramente, o Copilot realiza uma análise no ambiente de desenvolvimento onde o código está sendo escrito. Isso inclui verificar o que está sendo escrito antes e depois do cursor, quais arquivos estão abertos no editor de código, além das pastas e repositórios utilizados. Quando a solicitação do desenvolvedor ocorre por meio do *chat*, o Copilot acrescenta ainda mais informações ao seu entendimento de contexto — a inclusão do arquivo de código que está sendo editado, o trecho de código selecionado e todos os detalhes do projeto em geral, como os *frameworks*, linguagens de programação e bibliotecas.

Toda essa base contextual construída é enviada ao seu modelo de inteligência artificial. A partir disso, esse modelo usa cálculos probabilísticos que prevê quais trechos de código seriam mais adequados no momento e gera essas sugestões, cabendo ao desenvolvedor aceitar ou não.

3. Metodologia

Esta seção apresenta a metodologia adotada nesta revisão sistemática, contemplando tanto a definição da classificação do estudo quanto a descrição dos procedimentos metodológicos aplicados. São abordados o tipo e a natureza da pesquisa, bem como os critérios utilizados para seleção dos estudos, as bases de dados consultadas, os termos de busca aplicados e as etapas da triagem e análise. O objetivo é garantir clareza, transparência e reprodutibilidade, considerando que, segundo Nosek e Errington (2020), a replicação de um estudo — fundamental para a reprodutibilidade científica — depende diretamente do nível de detalhamento e da precisão com que o estudo original é descrito.

3.1 Classificação

Segundo Usman, Ali e Wohlin (2023), a definição clara do tipo e da abordagem metodológica da pesquisa é fundamental para assegurar o rigor científico e a consistência dos resultados em estudos secundários na área de engenharia de software.

Quanto à sua natureza, caracteriza-se como aplicada, pois analisa o crescimento do uso do GitHub Copilot no desenvolvimento de software e suas consequências.

Este estudo classifica-se como exploratório, com enfoque em investigar como a literatura científica tem relatado o uso do GitHub Copilot no processo de codificação e seus impactos na qualidade do código.

Foi adotada uma abordagem qualitativa, onde o foco ocorreu em aprofundar-se na análise do uso prático da ferramenta de IA e as percepções dos desenvolvedores em sua aplicação, a fim de compreender seu impacto tanto em cenários experimentais quanto sob a ótica dos próprios programadores.

Por fim, o procedimento adotado é a pesquisa bibliográfica, que se baseia na busca de materiais existentes a respeito do tema delimitado. Ela foi conduzida por meio de uma revisão sistemática da literatura, que permite uma seleção criteriosa e uma análise detalhada de estudos mais relevantes a respeito do tema de pesquisa.

3.2 Procedimento da pesquisa

Foi adotado o instrumento de intervenção *Knowledge Development Process - Constructivist* (ProKnow-C), um método de pesquisa, seleção e análise de artigos científicos com base no foco do pesquisador a fim de construir um portfólio bibliográfico que gere conhecimento a respeito do tema de pesquisa (Ensslin *et al.*, 2013, p. 333). Ressalta-se que para a elaboração do presente estudo, o uso do *ProKnow-C* foi adaptado. Algumas etapas foram seguidas integralmente, sendo estas: a definição do tema, a seleção de artigos por meio de critérios e a análise bibliométrica. Outras não foram aplicadas, como o teste de aderência de palavras-chave e o teste de representatividade, considerando que não seriam essenciais para os objetivos e escopo da pesquisa. Essa adaptação permitiu maior flexibilidade, sem comprometer o rigor metodológico, mantendo a transparência e reprodutibilidade do processo de análise e seleção dos estudos.

Diante disso, segue uma breve descrição das etapas adotadas:

1) Seleção do banco de artigos bruto: Onde, em primeiro lugar, se definem as palavras-chave e as bases de dados a serem feitas as buscas. Assim que são definidas, faz-se a busca de artigos nas bases de dados selecionadas com as palavras-chave e aplicação de filtros (recorte temporal, disponibilidade de acesso etc.).

2) Filtragem do banco de artigos bruto: Com uma amostra de artigos identificados nas bases de dados, faz-se uma filtragem que leva em consideração o alinhamento dos resumos e títulos de artigo com o tema e a sua disponibilidade de acesso aberto. Após isso, passa-se por uma nova triagem, desta vez aplicando os critérios de inclusão e exclusão. Por fim, faz-se a leitura integral dos artigos.

Ao seguir esse desenvolvimento, foi construído o portfólio bibliográfico final da pesquisa.

4. Resultados

Esta seção apresenta o detalhamento da aplicação do instrumento de intervenção *ProKnow-C,* bem como os principais resultados obtidos a partir dessa etapa da pesquisa. A abordagem permitiu a identificação e seleção dos estudos mais relevantes para os objetivos propostos, contribuindo para a construção de um portfólio bibliográfico alinhado com a temática investigada.

4.1 Seleção do portfólio bibliográfico

Etapa de formação de um conjunto de artigos alinhados com o tema da pesquisa. Se subdivide em outras duas fases:

4.1.1 Seleção do banco de artigos bruto

Definem-se os eixos de pesquisa, os quais, neste estudo, referem-se à relação entre a ferramenta GitHub Copilot e sua aplicação na codificação. Com isso definido, inicia-se outras três partes para a formação do conjunto de artigos bruto:

1) Definição de palavras-chave: Para facilidade de pesquisa acadêmica em documentos já publicados a respeito do assunto a ser trilhado, especifica-se palavras-chave a respeito dos eixos de pesquisa. Foram usados os termos:

a) *"software development"* e "Copilot" no intuito de encontrar estudos que relacionem o uso dessa IA no desenvolvimento de software;

b) *"review", "programming", "code", "suggestion", "autocomplete"* foram utilizados para especificar que os resultados de busca mais interessantes seriam aqueles que abordassem a fase de codificação;

c) por fim, *"quality"*, em busca de artigos que mencionem o impacto na qualidade.

2) Definição da base de dados: A construção do portfólio bibliográfico teve início com uma busca ampla e exploratória da literatura, realizada entre diferentes bases de dados e repositórios de acesso aberto. Foram utilizadas as bases disponibilizadas pelo Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), incluindo IEEE Xplore, ACM Digital Library e ScienceDirect. Por meio do Google Acadêmico, também foi possível encontrar estudos no repositório arXiv do Centro de Ciência da Computação da Universidade de Cornell.

3) Busca dos artigos nos bancos de dados com as palavras-chave: Com as palavras-chave e bases de dados definidas, aplica-se também o uso de operadores booleanos (AND, OR) e aplicação de filtros *"Open Access Only"* ou *"Artifacts Available"* para filtrar somente artigos de acesso aberto. A Tabela 1 retrata os resultados dessa busca:

Tabela 1 – Resultados de busca por base de dados.

Base de dados	Termos de busca	Nº de artigos encontrados	Filtro aplicado
ACM	Software development	4.187	*Artifacts Available*
ACM	Copilot	54	*Artifacts Available*
ACM	*autocomplete*	19	*Artifacts Available*
ACM	*ai software development*	784	*Artifacts Available*
Google Acadêmico	*copilot software development code OR review, OR quality, OR programming*	16.800	Sem filtro
Google Acadêmico	*copilot software development code OR review, OR quality, OR programming, OR code suggestion, OR autocomplete, OR IDE*	393	Sem filtro
IEEE	Software development	4.733	*Open Access Only*
IEEE	Copilot	8	*Open Access Only*
IEEE	*ai software development*	295	*Open Access Only*
IEEE	*Copilot; software*	86	Sem filtro
IEEE	*code; suggestion*	1.383	Sem filtro
ScienceDirect	*copilot, software development*	446	Sem filtro

Fonte: Elaborado pelas Autoras (2025).

As buscas resultaram em um total de 29.188 ocorrências nas bases de dados consultadas. No entanto, ressalta-se que esse número representa o volume bruto de resultados retornados pelas consultas. Para fins de triagem e avaliação manual, foram coletados apenas os títulos considerados mais alinhados aos objetivos da pesquisa, conforme leitura inicial e verificação da disponibilidade de acesso aos documentos. Essa coleta resultou na composição de uma planilha inicial contendo 86 registros, que serviram como ponto de partida para a etapa de seleção do portfólio bibliográfico.

4.1.2 Filtragem do banco de artigos bruto

Após a definição dos critérios de busca e a aplicação das palavras-chave nas bases selecionadas, foram coletados 86 registros considerados potencialmente relevantes para os objetivos desta revisão sistemática. Essa amostra inicial foi compilada manualmente, a partir da leitura dos títulos e resumos retornados nas buscas, considerando também a disponibilidade de acesso aos textos completos.

A seleção do portfólio seguiu os procedimentos sistemáticos pelo método *ProKnow-C*, incluindo a aplicação de critérios de inclusão e exclusão, com o intuito de garantir o alinhamento dos estudos ao escopo desta pesquisa. Os critérios utilizados estão definidos na Tabela 2.

Tabela 2 - Critérios de inclusão e exclusão de artigos.

Critérios de inclusão	Critérios de exclusão
Estudos acadêmicos publicados em periódicos ou conferências	Artigos não científicos ou que não tiveram revisão por pares
Publicações que tratam diretamente do GitHub Copilot	Estudos que não focam diretamente no GitHub Copilot
Foco na etapa de codificação e/ou qualidade do código	Estudos opinativos sem base empírica
Artigos em português ou inglês	Trabalhos que não tratam da qualidade do código ou codificação
Período entre 2021–2025	Período inferior a 2021

Fonte: Elaborado pelas Autoras (2025).

Embora a busca sistemática dos artigos tenha sido realizada em 2024, foi estabelecido como critério de inclusão o período até o ano de 2025, considerando possíveis atualizações e publicações oficiais de versões prévias. Nesse contexto, embora nenhum novo levantamento tenha sido feito em 2025, um dos estudos inicialmente localizado como *preprint* no repositório arXiv foi posteriormente publicado em periódico oficial no ano de 2025. Sendo assim, foi considerada a versão final do artigo, respeitando os critérios definidos previamente.

A aplicação desses critérios foi realizada de forma manual, com o apoio de uma planilha estruturada contendo colunas específicas para identificação da fonte, título, ano, foco temático, metodologia, tipo de estudo, presença de dados sobre qualidade de código, produtividade, percepções de desenvolvedores, entre outros.

Após a triagem completa, foram selecionados 15 artigos que compõem o portfólio bibliográfico desta revisão, apresentados na Tabela 3.

Tabela 3 - Portfólio bibliográfico dos artigos selecionados conforme os critérios.

Autores	Título do artigo	Ano	Periódico / Conferência	Nº de citações no Google Acadêmico
Peslak, A.; Koval.chick, L.	AI for Coders: An Analysis of the Usage of ChatGPT and GitHub Copilot	2024	Issues in Information Systems	1
Shi *et al.*	AI-Assisted Security: A Step towards Reimagining Software Development for a Safe Future	2023	2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)	6
Fagadau *et al.*	Analyzing Prompt Influence on Automated Method Generation	2024	ICPC '24: Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension	14

OBrien *et al.*	Are Prompt Engineering and TODO Comments Friends or Foes? An Evaluation on GitHub Copilot	2024	ICSE '24: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering	14
Pearce *et al.*	Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions	2022	2022 IEEE Symposium on Security and Privacy (SP)	566
Yetistiren *et al*.	Assessing the Quality of GitHub Copilot's Code Generation	2022	PROMISE 2022: Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering	167
Majdinasab *et al.*	Assessing the Security of GitHub Copilot's Generated Code - A Targeted Replication Study	2024	2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)	25
Zhang *et al.*	Demystifying Practices, Challenges and Expected Features of Using GitHub Copilot	2023	International Journal of Software Engineering and Knowledge Engineering	24
Vaithilingam *et al.*	Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models	2022	CHI EA '22: Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems	740
Barke *et al.*	Grounded Copilot: How Programmers Interact with Code-Generating Models	2023	Proceedings of the ACM on Programming Languages	423
Imai, Saki	Is GitHub Copilot a Substitute for Human Pair-programming? An Empirical Study	2022	2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)	182
Ziegler *et al.*	Measuring GitHub Copilot's Impact on Productivity	2024	Communications of the ACM	79
Idrisov, B.; Schillipe, Tim	Program Code Generation with Generative AIs	2024	Algorithms	50
Mozannar *et al.*	Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming	2024	CHI '24: CHI Conference on Human Factors in Computing Systems	138
Fu *et al.*	Security Weaknesses of Copilot Generated Code in GitHub	2025	ACM Transactions on Software Engineering and Methodology	49

Fonte: Elaborado pelas Autoras (2025).

5. Discussão

A seguir é apresentada a discussão dos resultados obtidos por meio da revisão sistemática da literatura, com o propósito de responder aos dois objetivos específicos delineados na pesquisa. A análise foi organizada em categorias que refletem os temas recorrentes identificados nos estudos selecionados. A discussão busca articular as evidências destacando as convergências e divergências, com o intuito de compreender o impacto do uso do GitHub Copilot na codificação.

5.1 Avaliação da qualidade do código gerado com o uso do GitHub Copilot

Esta seção apresenta os estudos que respondem ao objetivo específico: Investigar como a literatura científica tem avaliado a qualidade de código gerado com o uso do GitHub Copilot. Observou-se que essa temática tem sido explorada em diversas perspectivas, com enfoque em elementos que impactam diretamente a qualidade do código. Apresentam-se aqui as principais abordagens recorrentes em três tópicos: segurança, elaboração de *prompts* e manutenibilidade do código.

5.1.1 Segurança

Em relação ao aspecto de segurança do código gerado pelo GitHub Copilot, avaliada tanto em *setups* experimentais quanto em projetos reais hospedados na plataforma GitHub, observou-se que a ferramenta tem uma tendência a gerar códigos com potenciais vulnerabilidades.

Pearce *et al.* (2022) investigaram essa tendência fornecendo ao Copilot 89 prompts baseados nas vulnerabilidades catalogadas na *Common Weakness Enumeration* (CWE) Top 25 — uma lista de falhas de hardware e software que podem levar a vulnerabilidades de segurança —, com o objetivo de completar os programas e sendo posteriormente avaliados com a ferramenta CodeQL e inspeção manual. Constatou-se que 40% das sugestões geradas foram consideradas vulneráveis. Também foi apresentado que pequenas mudanças no *prompt* influenciam a segurança do código e que a qualidade das sugestões pode variar dependendo do nível de base de treinamento, como ocorreu na baixa qualidade de sugestão para Verilog.

A replicação do estudo, feita por Majdnissab *et al.* (2024), voltada para testar as atualizações de segurança do GitHub Copilot e com foco exclusivo na linguagem de programação Python, apresentou uma redução percentual de 27,25% da taxa de sugestões de códigos vulneráveis, em comparação com 35,54% de performance da mesma linguagem apresentada na pesquisa original, entretanto ainda foram percebidas algumas falhas em determinados cenários.

Semelhantemente, o estudo de Shi *et al.* (2023) seguiu esse foco de avaliação de segurança, porém restringindo-se a dois cenários: *SQL Injection* e *Hard-Coded Credentials*, utilizando comentários explícitos que instruíam o que o Copilot deveria sugerir e se deveria ou não conter vulnerabilidade. Identificou-se que esses comentários influenciam fortemente nas sugestões geradas pela IA.

Quanto ao estudo de Fu *et al.* (2025), desta vez voltado à avaliação dos códigos em projetos reais que utilizaram o GitHub Copilot e outras IAs, com o uso de ferramentas de análise estática (CodeQL, FindBugs/SpotBugs, ESLint, Bandit, and GoSec), detectou vulnerabilidades associadas a *CWEs*. Além disso, foi avaliado o uso do Copilot Chat em sua capacidade de consertar as falhas, e, embora vantajosa, sua eficácia varia de acordo com o tipo de vulnerabilidade lidada.

A justificativa comum dos autores a respeito da origem das falhas se embasa no fato de que o GitHub Copilot tem sua fonte de dados provenientes de uma ampla base de código aberto, que pode conter padrões inseguros. Também entram em concordância da necessidade de uma revisão humana e o uso de ferramentas de verificação que garantam a segurança, mediante a tendência de geração de trechos com potenciais vulnerabilidades.

5.1.2 Elaboração de *prompts*

Foi notório, em alguns estudos, o impacto da estrutura de um *prompt* no resultado de código. O estudo de Yetistiren *et al.* (2022) analisou essa influência fornecendo tarefas de programação advindas do *HumanEval dataset* para o GitHub Copilot resolver, tendo como critério de avaliação: sintaxe, correção e eficiência. O resultado foi que o Copilot é capaz de gerar códigos com sintaxes corretas e desempenho eficiente, mas houve uma queda de assertividade na resolução de testes unitários com a retirada de informações contextuais das *docstrings* e funções descritivas.

O estudo de Fagadau *et al.* (2024) foi conduzido de uma maneira mais abrangente, testando 124.800 combinações de prompts a partir de 200 métodos Java e fornecidos ao GitHub Copilot para que este completasse o corpo do método. Os resultados apresentam que *prompts* que incluíam exemplos e resumos do método, escritos no tempo presente, estes sendo mais bem interpretados pelo Copilot do que o tempo futuro, levaram a códigos mais corretos e funcionais. Em contrapartida, *prompts* com casos limite ou informações excessivas contextuais não melhoraram a qualidade e, em alguns casos, até a reduziram.

Quanto à escrita de prompts para tratamento de códigos com dívida técnica, a pesquisa de Obrien *et al*. (2024) buscou responder se o uso dos comentários *"TODO"* no prompt poderiam ser aliados de orientação à IA para resolvê-los ou se potencializavam a mitigação de códigos endividados. A análise de 1.140 códigos gerados a partir de 380 comentários indicou

que os fatores positivos para elaboração de soluções úteis estão na remoção da palavra *"TODO",* comentários que descrevem ações concretas, informações contextuais, justificativas e considerações futuras. A inclusão direta dos comentários *TODO*, textos com questionamentos e que descrevem somente os sintomas dificultam a resolução do Copilot.

Nesse sentido, entende-se que a eficiência é uma via de mão dupla, onde o resultado não reside somente na base de conhecimento que o Copilot possui, mas também na forma do desenvolvedor elaborar a sua solicitação à ferramenta.

5.1.3 Manutenibilidade

Ao analisar a manutenibilidade dos códigos gerados por IAs — dentre elas o GitHub Copilot — comparando com a dos seres humanos, o estudo de Idrisov e Schillipe (2024) destacou que o Copilot foi a ferramenta que mais apresentou soluções corretas para os problemas de codificação propostos, superando as outras IAs avaliadas, embora não tenha alcançado a taxa de acerto dos programadores humanos em todas as tarefas.

Em relação ao desempenho em métricas de qualidade, os resultados foram variados: O Copilot superou os humanos em linhas de código, complexidade ciclomática, complexidade de espaço e índice de manutenibilidade, mas demonstrou ser inferior em tempo de execução e complexidade de tempo. Ademais, o estudo também sugere que, mesmo quando o código do GitHub Copilot está incorreto, ele pode servir como base útil para ser corrigido manualmente do que programar do zero.

5.2 Percepção dos desenvolvedores quanto ao uso do Copilot

Esse objetivo específico consiste em analisar as percepções dos desenvolvedores quanto ao impacto do GitHub Copilot em sua produtividade e no processo de codificação. É essencial compreender essa visão, pois o uso das ferramentas de inteligência artificial e seus resultados também são afetadas pelas experiências dos profissionais. Portanto, apresenta-se neste bloco, os principais achados dos estudos analisados, falando a respeito das vantagens percebidas, desafios encontrados e a confiança atribuída ao Copilot durante o processo de desenvolvimento.

5.2.1 Perfil

A ferramenta é usada por desenvolvedores de diversos perfis, desde estudantes até programadores sêniores, entretanto, a partir dos resultados da análise de dados feita por Peslak e Kovalchick (2024) da pesquisa global *Stack Overflow* 2023, foi apontado que a maior taxa de uso do GitHub Copilot está nos iniciantes em programação e naqueles que programam como *hobby*.

Zhang *et al.* (2023), ao analisar as discussões públicas de *Stack Overflow* e *GitHub Discussions*, identificaram que a principal motivação de os desenvolvedores aderirem à ferramenta é pela geração automática de código. Entre os benefícios relatados no uso da ferramenta, destaca-se o desenvolvimento mais rápido, fornecimento de ideias de código e a adaptação aos padrões de código dos usuários. Por outro lado, foram mencionadas as limitações, sendo a principal a dificuldade de integração com outras IDEs e *plugins*.

5.2.2 Confiança excessiva

Alguns estudos demonstraram que a confiança excessiva depositada na ferramenta afeta o fluxo de trabalho do desenvolvedor, levando-o a um retrabalho para validação ou reescrita do código.

Isso foi notável no estudo de Imai (2022), o qual fez um experimento empírico com 21 programadores, propondo-lhes o desenvolvimento de um projeto de código, tendo um humano como dupla ou o GitHub Copilot para auxiliar, a fim de comparar a produtividade e qualidade do código, usando como métrica a quantidade de linhas. Embora a ferramenta de IA tenha gerado mais linhas, a qualidade foi baixa. Além disso, com análise de *eye tracking*, o autor levanta a hipótese de que, acompanhados com uma IA, os desenvolvedores inspecionam menos o código gerado por ela, mediante a alta confiança depositada na

ferramenta.

Barke *et al.* (2023) avaliou a interação dos programadores que já tiveram uma experiência prévia com o GitHub Copilot. Por meio da observação, os pesquisadores identificaram dois padrões principais de interação dos desenvolvedores com a ferramenta: modo de aceleração (adotadas por aqueles que já têm em mente o que precisa fazer) e modo de exploração (usado na situação em que o programador não tem certeza de como proceder em tarefas novas ou desconhecidas (o GitHub Copilot explora as opções para avaliar alternativas). Como resultado, foi observado que, por parte daqueles que optaram pelo modo de exploração, o fizeram impulsionados por uma confiança excessiva na ferramenta. Os programadores passaram mais tempo no modo de exploração do que no modo de aceleração, e isso se deve à necessidade de uma validação minuciosa, que inclui: revisão e execução do código, consulta à documentação e uso de ferramenta de análise estática.

Seguindo a mesma linha de análise, o estudo de Mozannar *et al.* (2024) investigou os comportamentos dos programadores ao interagirem com as sugestões de código fornecidas pelo GitHub Copilot. Para tanto, foi elaborada uma taxonomia denominada *CodeRec User Programming States* (CUPS), a qual cataloga 12 atividades dos profissionais ao interagirem com a IA (escrevendo novo código, testando o código, elaborando prompts etc.). Os dados foram obtidos através de uma análise retrospectiva das sessões de codificação de 21 programadores, as quais foram gravadas em vídeo, e em seguida associaram manualmente cada parte do processo de codificação a um estado do CUPS.

Os principais resultados apresentam que a atividade de verificação de sugestões do Copilot foi a que mais consumiu tempo dos participantes. Também foi notável que os desenvolvedores frequentemente entraram no estado de "Adiar o Pensamento Para Mais Tarde" e tal comportamento implicava no aumento das taxas de aceitação das sugestões, embora estas exigissem verificação e edição subsequentes. Os autores apresentam que estados específicos de interação com a IA somaram, em média, 51,5% do tempo da sessão, o que aponta para um impacto direto no fluxo de trabalho do desenvolvedor.

5.2.3 Produtividade

Vaithilingam *et al.* (2022) conduziram um estudo experimental com 24 participantes para comparar o GitHub Copilot ao *IntelliSense* — recurso de autocompletar tradicional utilizado nas IDEs, que sugere palavras-chave, variáveis e trechos de código com base na sintaxe da linguagem — em tarefas de programação, avaliando as percepções dos desenvolvedores na questão da usabilidade e o modo de lidar com os erros. A maioria dos usuários relatou a preferência pelo Copilot, justificando-se por perceberem a vantagem de utilizá-lo como ponto de partida e ferramenta de apoio. Entretanto, também foi apresentado que muitos tiveram dificuldades em compreender e editar o código gerado, especialmente em tarefas complexas. Outro obstáculo para os programadores também foi nos trechos de código longos gerados, o que causou uma sobrecarga cognitiva e dificuldade de depuração, alguns participantes relatam também ter sentido perder o controle do código e usaram o *IntelliSense*.

O estudo de Ziegler *et al.* (2024) combinou os dados de telemetria com um questionário respondido por mais de 2 mil desenvolvedores, em prol de investigar a produtividade percebida com o uso do Copilot. Os resultados indicaram que a métrica principal de produtividade está correlacionada com a taxa de aceitação das sugestões, principalmente entre os iniciantes. Os desenvolvedores mais experientes relataram benefícios na automatização de tarefas repetitivas e apoio em lidar com novas linguagens.

6. Conclusão

Diante do contexto apresentado sobre a tendência de investimento da Inteligência Artificial nas organizações nos próximos anos e correlacionando essa ferramenta ao processo de desenvolvimento de software visando o aumento da qualidade, este presente estudo teve como objetivo analisar como a literatura científica tem avaliado o GitHub Copilot como uma solução alternativa para melhorar a qualidade do produto de software, especialmente na fase de codificação. Foi exposto que o Copilot

atua como um "copiloto" para os desenvolvedores por conta de suas capacidades atuais que incluem sugerir códigos em tempo real, interagir por meio de chat para esclarecimento de dúvidas de programação e oferecer suporte via linha de comando.

Diante dos estudos selecionados nesta revisão sistemática, pode-se concluir que a ferramenta é útil como apoio para os desenvolvedores, destacando-se em ser um bom ponto de partida e direcionamento na programação. Contudo, pode-se considerar que é uma ferramenta ainda precoce e não recomendável de ser usada isoladamente, levando em consideração seu potencial quanto à geração de código inseguro e dificuldade em lidar com tarefas complexas.

Ademais, observa-se que os resultados também são influenciados pela maneira como os desenvolvedores utilizam a IA. A confiança excessiva, a ausência de uma reflexão mais prolongada quanto às sugestões de código e um *prompt* mal estruturado pode ocasionar retrabalho, especialmente entre os principiantes na programação, que são o maior público de pessoas que utilizam o GitHub Copilot.

Em suma, espera-se que esse estudo possa ter colaborado para o fornecimento de informações provenientes de estudos científicos a respeito do GitHub Copilot, com informações acerca da ferramenta que impulsionem sua adoção para o processo de codificação. Almeja-se que este trabalho possa inspirar futuras pesquisas que apresentem a evolução contínua desta tecnologia e suas contribuições no desenvolvimento de software.

Agradecimentos

À professora Cristina, pela orientação na construção deste trabalho e pelo incentivo à sua submissão a uma revista, acreditando em seu potencial.

À banca examinadora, pelas contribuições, sugestões e apontamentos que contribuíram para o aprimoramento deste trabalho.

A todos que direta ou indiretamente contribuíram para a realização e sucesso deste artigo.

Referências

Barke, S., James, M. B., & Polikarpova, N. (2023). Grounded Copilot: How Programmers Interact with Code-Generating Models. Proceedings of the ACM on Programming Languages, 7(OOPSLA1), 85–111. https://doi.org/10.1145/3586030

Baskhad Idrisov, & Schlippe, T. (2024). Program Code Generation with Generative AIs. Algorithms, 17(2), 62–62. https://doi.org/10.3390/a17020062

ChatGPT, a inteligência artificial como você nunca viu, é a próxima revolução | Brasil. (2023, February 24). McKinsey & Company. https://www.mckinsey.com/br/our-insights/all-insights/chatgpt-e-a-revolucao-da-inteligencia-artificial?form=MG0AV3

Ensslin, L., Ensslin, S. R., & Pinto, H. de M. (2013). Processo de investigação e análise bibliométrica: avaliação da qualidade dos serviços bancários. Revista de Administração Contemporânea, 17(3), 325–349. https://doi.org/10.1590/s1415-65552013000300005

Fagadau, I. D., Mariani, L., Micucci, D., & Riganelli, O. (2024, February 13). Analyzing Prompt Influence on Automated Method Generation: An Empirical Study with Copilot. ArXiv.org. https://doi.org/10.1145/3643916.3644409

Fu, Y., Liang, P., Tahir, A., Li, Z., Shahin, M., Yu, J., & Chen, J. (2025). Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study. ACM Transactions on Software Engineering and Methodology. https://doi.org/10.1145/3716848

GitHub. (2025). GitHub Copilot · Your AI pair programmer. GitHub. https://GitHub.com/features/copilot

GitHub. (2025). What is GitHub Copilot? GitHub Docs. https://docs.GitHub.com/en/copilot/about-GitHub-copilot/what-is-GitHub-copilot

Hussein Mozannar, Bansal, G., Fourney, A., & Horvitz, E. (2024). Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming. https://doi.org/10.1145/3613904.3641936

IDC - About - Home. (2019). IDC: The Premier Global Market Intelligence Company. https://www.idc.com/about

Imai, S. (2022, May 1). Is GitHub Copilot a Substitute for Human Pair-programming? An Empirical Study. IEEE Xplore. https://doi.org/10.1145/3510454.3522684

Introducing ChatGPT. (2022, November 30). OpenAI. https://openai.com/index/chatgpt/

Jyoti, R., & Schubmehl, D. (2024). Business opportunity of AI: Generative AI adoption and business impact. International Data Corporation (IDC). Recuperado de https://info.microsoft.com/ww-landing-business-opportunity-of-ai.html

Krasner, H. (2022). The cost of poor quality software in the US: A 2022 report. Consortium for Information & Software Quality. Recuperado de https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/

Lopes, A. (2023). Introdução aos LLMs e à IA generativa. BRAINS. https://brains.dev/2023/introducao-aos-llms-e-a-ia-generativa/

Nosek, B. A., & Errington, T. M. (2020). What Is Replication? PLOS Biology, 18(3). https://doi.org/10.1371/journal.pbio.3000691

OBrien, D., Biswas, S., Sayem Mohammad Imtiaz, Rabe Abdalkareem, Emad Shihab, & Rajan, H. (2024). Are Prompt Engineering and TODO Comments Friends or Foes? An Evaluation on GitHub Copilot. https://doi.org/10.1145/3597503.3639176

Paula, J. de. (2024, April 5). Dívida Técnica: como reconhecer, entender e superar. Objective. https://www.objective.com.br/insights/divida-tecnica/

Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022, May 1). Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. IEEE Xplore. https://doi.org/10.1109/SP46214.2022.9833571

Peslak, A., & Kovalchick, L. (2024). AI for coders: An analysis of the usage of ChatGPT and GitHub Copilot. Issues in Information Systems. https://iacis.org/iis/2024/4_iis_2024_252-260.pdf

Pressman, R. S., & Maxim, B. R. (2021). Engenharia de software: uma abordagem profissional (9ª ed.). AMGH.

Priberam Informática, S.A. (2024). Dicionário Priberam da Língua Portuguesa. Dicionário Priberam Da Língua Portuguesa. https://dicionario.priberam.org/revolucion%C3%A1rio

RocketCode. (2023, September 23). Entendendo a dívida técnica no desenvolvimento de software. https://rocketcode.com.br/blog/entendendo-a-divida-tecnica-no-desenvolvimento-de-software/

Sauvola, J., Tarkoma, S., Klemettinen, M., Riekki, J., & Doermann, D. (2024). Future of software development with generative AI. Automated Software Engineering, 31(1). https://doi.org/10.1007/s10515-024-00426-z

Sena, J., Barreto, A., Barbosa, J., & Alves, K. (2024). POTENCIALIDADES E DESAFIOS DO GitHub COPILOT COMO FERRAMENTA DA INTELIGÊNCIA ARTIFICIAL. P2P E INOVAÇÃO, 10(2). https://doi.org/10.21728/p2p.2024v10n2e-7031

Shi, Y., Nazmus Sakib, Hossain Shahriar, Lo, D., Chi, H., & Qian, K. (2023). AI-Assisted Security: A Step towards Reimagining Software Development for a Safer Future. https://doi.org/10.1109/compsac57700.2023.00142

Song, F., Agarwal, A., & Wen, W. (2024). The Impact of Generative AI on Collaborative Open-Source Software Development: Evidence from GitHub Copilot. ArXiv.org. https://arxiv.org/abs/2410.02091

Stack Overflow. (2024). 2024 developer survey: AI. https://survey.stackoverflow.co/2024/ai/

Usman, M., Bin Ali, N., & Wohlin, C. (2023). A Quality Assessment Instrument for Systematic Literature Reviews in Software Engineering. E-Informatica Software Engineering Journal, 17(1), 230105. https://doi.org/10.37190/e-inf230105

Vahid Majdinasab, Bishop, M. J., Rasheed, S., Arghavan Moradidakhel, Tahir, A., & Foutse Khomh. (2024). Assessing the Security of GitHub Copilot's Generated Code - A Targeted Replication Study. https://doi.org/10.1109/saner60148.2024.00051

Vaithilingam, P., Zhang, T., & Glassman, E. L. (2022). Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. CHI Conference on Human Factors in Computing Systems Extended Abstracts. https://doi.org/10.1145/3491101.3519665

Yetistiren, B., Ozsoy, I., & Tuzun, E. (2022). Assessing the quality of GitHub copilot's code generation. Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering. https://doi.org/10.1145/3558489.3559072

Zhang, B., Liang, P., Zhou, X., Ahmad, A., & Waseem, M. (2023). Demystifying Practices, Challenges and Expected Features of Using GitHub Copilot. International Journal of Software Engineering and Knowledge Engineering, 1–20. https://doi.org/10.1142/s0218194023410048

Ziegler, A., Eirini Kalliamvakou, X. Alice Li, Rice, A., Rifkin, D., Simister, S., Ganesh Sittampalam, & Aftandilian, E. (2024). Measuring GitHub Copilot's Impact on Productivity. Communications of the ACM, 67(3), 54–63. https://doi.org/10.1145/3633453