

UNIVERSIDADE DE SÃO PAULO  
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA  
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**Percepções em Transformação**

*Os Impactos da IA Generativa na Produção  
de Software.*

Cássio Azevedo Cancio

**MONOGRAFIA FINAL**

**MAC 499 — TRABALHO DE  
FORMATURA SUPERVISIONADO**

Supervisor: Paulo Meirelles

Cossupervisor: Arthur Pilone

*O conteúdo deste trabalho é publicado sob a licença CC BY 4.0  
(Creative Commons Attribution 4.0 International License)*

*Aos meus pais, que sempre incentivaram meus estudos.  
Aos meus professores, que tornaram este trabalho possível.*



## Resumo

Cássio Azevedo Cancio. **Percepções em Transformação: Os Impactos da IA Generativa na Produção de Software.** Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo.

Este trabalho visa reunir e analisar dados sobre os impactos da IA generativa na produção de *software*, focando na evolução da percepção dos programadores e comparando diferentes pesquisas e relatórios sobre o tema. A metodologia adotada envolveu a seleção e a análise de relatórios de mercado de 2023 a 2025 e de revisões recentes da literatura. Os resultados demonstram um crescimento significativo na adoção de ferramentas de IA por desenvolvedores profissionais, atingindo 80,7%, em uma das pesquisas analisadas. No entanto, essa adoção foi acompanhada por uma queda na percepção de qualidade e no nível de confiança dos profissionais sobre os resultados gerados pela IA. Paralelamente, uma análise de métricas de código, como o aumento do churn e da duplicação, e a queda da refatoração, sugere uma deterioração da qualidade do código. O estudo também aponta uma discrepância entre o uso predominante da IA para escrita de código, considerada a atividade mais agradável pelos desenvolvedores, e o desejo declarado de delegar tarefas menos prazerosas, como a escrita de testes. O cenário atual indica a continuidade da expansão da IA na engenharia de *software*, especialmente com a ascensão da IA agêntica, o que reforça a necessidade de novos estudos para mitigar os riscos associados à qualidade do código.

**Palavras-chave:** IA Generativa. Engenharia de Software. Assistente de Código de IA.



# Abstract

Cássio Azevedo Cancio. **Perceptions in Transformation: Impacts of AI on Code**

**Production.** Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo.

This work aims to gather and analyze data on the impacts of generative AI on software development, focusing on the evolution of developers' perceptions and comparing different research studies and reports on the subject. The adopted methodology involved the selection and analysis of market reports from 2023 to 2025, as well as recent literature reviews. The results show a significant increase in the adoption of AI tools by professional developers, reaching 80.7% in one of the analyzed studies. However, this adoption was accompanied by a decline in perceived quality and in the level of confidence professionals place in AI-generated outputs. In parallel, an analysis of code metrics—such as increased churn and duplication, and a decrease in refactoring—suggests a deterioration in code quality. The study also highlights a discrepancy between the predominant use of AI for code writing, considered the most enjoyable activity by developers, and the expressed desire to delegate less pleasurable tasks, such as writing tests. The current scenario indicates the continued expansion of AI in software engineering, especially with the rise of agentic AI, reinforcing the need for further studies to mitigate risks associated with code quality.

**Keywords:** Generative AI. Software Engineering. AI Code Assistant.



# Listas de abreviaturas

ANSSI	<i>Agence Nationale de la Sécurité des Systèmes d'Information</i> (Agência Francesa de Segurança dos Sistemas de Informação)
API	<i>Application Programming Interface</i> (Interface de Programação de Aplicações)
BSI	<i>Bundesamt für Sicherheit in der Informationstechnik</i> (Escritório Alemão de Segurança da Informação)
CIO	<i>Chief Information Officer</i> (Diretor de Tecnologia da Informação)
IA	Inteligência Artificial
IDE	<i>Integrated Development Environment</i> (Ambiente de Desenvolvimento Integrado)
IME	Instituto de Matemática e Estatística
LLM	<i>Large Language Model</i> (Modelo de Linguagem de Grande Escala)
OTAN	Organização do Tratado do Atlântico Norte
RNF	Requisitos Não-Funcionais
SDD	<i>Software Design Document</i>
SDLC	<i>Software Development Life Cycle</i> (Ciclo de Desenvolvimento de Software)
SRS	<i>Software Requirements Specification</i>
USP	Universidade de São Paulo

# Listas de figuras

3.1	Ferramentas utilizadas pelos desenvolvedores (SERGEYUK <i>et al.</i> , 2025; n = 481)	16
3.2	Gráfico representando quais tarefas do desenvolvimento os desenvolvedores gostam de fazer, quais eles delegariam à IA e qual sua impressão de dificuldade sobre as atividades (SERGEYUK <i>et al.</i> , 2025) . . . . .	16
3.3	Proporção anual das linhas modificadas classificadas pelo tempo que permaneceram no repositório antes de sofrerem uma nova modificação significativa entre 2020 e 2024 (GITCLEAR, 2025). . . . .	19
3.4	Proporção anual dos desenvolvedores profissionais que utilizam ferramentas de IA em seu processo de desenvolvimento entre 2023 e 2025 (STACK OVERFLOW, 2023; 2024; 2025; n <sub>2023</sub> = 67237; n <sub>2024</sub> = 46049; n <sub>2025</sub> = 33662). . . . .	20
3.5	Proporção anual da percepção dos desenvolvedores profissionais em relação às ferramentas de IA de 2023 a 2025 (STACK OVERFLOW, 2023; 2024; 2025; n <sub>2023</sub> = 46928; n <sub>2024</sub> = 35142; n <sub>2025</sub> = 25814). . . . .	20
3.6	Proporção anual do nível de confiança dos desenvolvedores em ferramentas de IA entre 2023 e 2025 (STACK OVERFLOW, 2023; 2024; 2025; n <sub>2023</sub> = 39042; n <sub>2024</sub> = 28829; n <sub>2025</sub> = 25701). . . . .	21
3.7	Percepção dos desenvolvedores profissionais sobre a capacidade das ferramentas de IA em lidar com tarefas complexas em 2025 (STACK OVERFLOW, 2025; n = 25695). . . . .	21
3.8	Uso atual e interesse futuro no uso de ferramentas de IA ao longo do fluxo de desenvolvimento em 2024 (STACK OVERFLOW, 2024; n = 35978) . . . . .	22
3.9	Quadrante Mágico dos assistentes de código com IA (BATCHU <i>et al.</i> , 2024)	24
3.10	Principais resultados esperados por CIOs nos negócios com a aplicação da IA generativa (COSHOW <i>et al.</i> , 2024) . . . . .	26

# Listas de tabelas

3.1	Temas resultantes nas respostas à pergunta "O que impede você de usar assistentes de IA nas etapas do fluxo de trabalho que você não os utiliza IA?"(SERGEYUK <i>et al.</i> , 2025). . . . .	15
3.2	Histórico de métricas de código de 2020 a 2024 (GITCLEAR, 2025). . . . .	18
3.3	Histórico de duplicação de código nos commits de 2020 a 2024 (GITCLEAR, 2025). . . . .	18
3.4	Proporção de desenvolvedores que não têm interesse em utilizar ferramentas de IA em diferentes partes do SDLC entre 2023 e 2025. As atividades presentes em apenas uma das pesquisas não foram incluídas (STACK OVERFLOW, 2023; 2024; 2025; n <sub>2023</sub> = 37726; n <sub>2024</sub> = 35978; n <sub>2025</sub> = 25349). . . . .	23



# Sumário

<b>Introdução</b>	<b>1</b>
Contexto . . . . .	1
Objetivos . . . . .	2
Estrutura do trabalho . . . . .	2
<b>1 Referencial teórico</b>	<b>3</b>
1.1 Engenharia de <i>software</i> . . . . .	3
1.1.1 Etapas do desenvolvimento de <i>software</i> . . . . .	3
1.1.2 Ferramentas de desenvolvimento . . . . .	6
1.2 Inteligência artificial . . . . .	6
1.2.1 Aprendizado de máquina . . . . .	7
1.2.2 Aprendizado profundo . . . . .	7
1.2.3 IA generativa . . . . .	7
1.2.4 Modelos de linguagem de grande escala (LLMs) . . . . .	8
<b>2 Estratégia e seleção das fontes</b>	<b>9</b>
2.1 Seleção das fontes . . . . .	9
2.1.1 Literatura formal . . . . .	9
2.1.2 Literatura cinzenta . . . . .	10
2.2 Análises . . . . .	11
<b>3 Fontes</b>	<b>13</b>
3.1 Literatura formal . . . . .	13
3.1.1 “Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward” (2025) . . . . .	13
3.2 Literatura cinzenta . . . . .	17
3.2.1 <i>AI Copilot Code Quality: Evaluating 2024’s Increased Defect Rate via Code Quality Metrics</i> (2025) . . . . .	17
3.2.2 <i>Stack Overflow Developer Survey</i> (2023; 2024; 2025) . . . . .	19

3.2.3	<i>Magic Quadrant for AI Code Assistants (2024)</i>	22
3.2.4	<i>Top Strategic Technology Trends for 2025: Agentic AI (2024)</i>	25
3.2.5	<i>AI Coding Assistants (2024)</i>	27
<b>4</b>	<b>Análise</b>	<b>29</b>
4.1	Amadurecimento da percepção dos programadores	29
4.2	Indícios da adoção e extração	29
4.3	Desconexão entre discurso e prática	30
4.4	Tendência de automatização	30
4.5	As métricas e o futuro	30
<b>5</b>	<b>Conclusão</b>	<b>31</b>
<b>Referências</b>		<b>33</b>

# Introdução

## Contexto

A engenharia de *software* é um campo da computação que se dedica à produção e à manutenção de sistemas de *software*. O termo foi consolidado em 1968 pela Organização do Tratado do Atlântico Norte (OTAN), no contexto da chamada “crise do *software*”, período marcado pelo aumento da complexidade dos sistemas computacionais e pela dificuldade em desenvolver programas de forma previsível e organizada. Desde então, a área passou por um processo contínuo de amadurecimento, com o surgimento de ferramentas, métodos e processos voltados à organização do desenvolvimento de *software* e à viabilização de projetos cada vez mais complexos.

Nas décadas seguintes, avanços tecnológicos significativos contribuíram para transformar o cenário computacional, incluindo o aumento da capacidade de processamento, o barateamento do *hardware*, tornando computadores e dispositivos móveis mais acessíveis, e a expansão da *internet*, que possibilitou a inclusão de bilhões de pessoas em ambientes digitais. Esse processo resultou na geração de grandes volumes de dados sobre diferentes aspectos da vida cotidiana e virtual.

Nesse contexto, técnicas de aprendizado de máquina puderam evoluir de forma acelerada, culminando no desenvolvimento de modelos capazes de aprender padrões complexos a partir de grandes conjuntos de dados. Entre esses avanços, destacam-se os modelos de inteligência artificial generativa, em especial os modelos de linguagem de grande escala (LLMs), capazes de gerar conteúdos novos, como texto e código, a partir de padrões aprendidos durante seu treinamento.

Com o tempo, a aplicação de modelos de IA generativa em ferramentas usadas na engenharia de *software* tornou-se cada vez mais popular. As ferramentas incluem *chatbots* conversacionais utilizados para auxílio na escrita e compreensão de código, assistentes de programação integrados a ambientes de desenvolvimento integrados (IDEs), ferramentas para geração de testes e suporte à depuração.

Nos últimos anos, diversos estudos passaram a investigar o uso dessas ferramentas no processo de desenvolvimento, analisando seus impactos, limitações e riscos, bem como propondo diretrizes para uma adoção segura e responsável (JOHNSON e MENZIES, 2024 e TERRAGNI *et al.*, 2025). Dados recentes indicam que a adoção dessas tecnologias já é expressiva entre desenvolvedores profissionais. Segundo uma pesquisa conduzida pela STACK OVERFLOW (2025), 80,7% dos desenvolvedores profissionais participantes utilizam

ferramentas de inteligência artificial em seu processo de desenvolvimento de *software*, enquanto 4,6% afirmam que pretendem adotá-las em breve.

Este trabalho se propõe a investigar os impactos do uso de ferramentas de IA generativa na engenharia de *software*, reunindo e analisando dados provenientes tanto da literatura formal quanto da literatura cinzenta. Considerando a ampla disseminação da IA na engenharia de *software*, é razoável supor que seu uso produza impactos relevantes na produção de código e nas práticas de desenvolvimento, o que reforça a relevância deste trabalho.

## Objetivos

Os objetivos deste trabalho são:

- Reunir e sistematizar dados sobre os impactos do uso de ferramentas de IA generativa na produção de *software*, considerando tanto evidências técnicas quanto percepções de profissionais da área;
- Analisar a evolução da percepção dos programadores em relação ao uso dessas ferramentas ao longo do tempo, identificando tendências, benefícios percebidos e preocupações recorrentes;
- Comparar e contrastar os resultados provenientes da literatura formal e da literatura cinzenta sobre o tema, com o objetivo de identificar convergências, divergências e lacunas entre pesquisas acadêmicas e evidências oriundas do mercado;
- Sintetizar os principais achados da análise, discutindo suas implicações para a prática do desenvolvimento de *software* e para pesquisas futuras na área.

## Estrutura do trabalho

O [Capítulo 1](#) é uma fundamentação teórica, apresentando os conceitos essenciais para o trabalho. O [Capítulo 2](#) descreve a estratégia utilizada para a seleção das fontes e a organização das análises. No [Capítulo 3](#), são apresentadas análises de cada uma das fontes do trabalho, com um resumo dos resultados e da organização por trás das fontes. O [Capítulo 4](#) contém uma análise que compara e relaciona os estudos das duas frentes. Por fim, o [Capítulo 5](#) conclui o trabalho, apontando possíveis continuidades.

# Capítulo 1

## Referencial teórico

### 1.1 Engenharia de *software*

Segundo a definição de *IEEE Standard Glossary of Software Engineering Terminology* (1990), a engenharia de *software* consiste na aplicação de uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção de *software*. Diante da complexidade dos sistemas atuais, foram criadas etapas, metodologias e ferramentas para apoiar os atores envolvidos no projeto, como desenvolvedores, analistas, investidores e clientes.

#### 1.1.1 Etapas do desenvolvimento de *software*

O Ciclo de Vida de Desenvolvimento de Software (*Software Development Life Cycle* ou SDLC) é composto por uma sequência de processos que visam produzir um *software* eficaz e de alta qualidade. Embora haja variações no número de etapas de acordo com diferentes fontes, em geral, são consideradas sete fases essenciais: planejamento, análise de requisitos, *design*, codificação, testes, implantação e manutenção.

##### Planejamento

A fase inicial envolve definir o propósito e o escopo do *software*. Durante essa etapa, a equipe de desenvolvimento levanta as tarefas necessárias, elabora estratégias para realizá-las e colabora para compreender as necessidades dos usuários finais. Os objetivos do *software* devem ficar claros para todos os envolvidos.

Além disso, ocorre o estudo de viabilidade, no qual desenvolvedores e demais atores do projeto avaliam desafios técnicos e financeiros que possam impactar a evolução e o sucesso do sistema. Ao final desta fase, é criado um plano de projeto, detalhando as funções do sistema, os recursos necessários, possíveis riscos e o cronograma de execução. Ao definir papéis, responsabilidades e expectativas, o planejamento estabelece uma base sólida para um processo eficiente de desenvolvimento.

## Análise de requisitos

Nesta etapa, a equipe de projeto realiza o levantamento dos requisitos por meio da coleta de informações das partes interessadas, como analistas, usuários e clientes. São empregadas técnicas como entrevistas, pesquisas e grupos de foco para compreender as necessidades e expectativas dos usuários.

Após a coleta, os dados são analisados, diferenciando os requisitos essenciais dos desejáveis. Essa análise permite definir as funcionalidades, desempenho, segurança e interfaces do *software*. Nesse momento, são estabelecidos os requisitos funcionais e não funcionais. Os requisitos funcionais especificam as funções que o *software* deve possuir, enquanto os requisitos não funcionais tratam de como o sistema deve se comportar, incluindo aspectos como desempenho, segurança, usabilidade e escalabilidade.

O resultado desse processo é o Documento de Especificação de Requisitos (DER), também referido na literatura como *Software Requirements Specification* (SRS), que descreve o propósito, as funcionalidades e características do *software*, servindo como guia para a equipe de desenvolvimento e fornecendo estimativas de custo. O êxito desta fase é crucial para o sucesso do projeto, pois garante que a solução desenvolvida atenda às expectativas dos usuários.

## Design

A fase de *design* é responsável pela definição da estrutura do *software*, abrangendo sua funcionalidade e aparência. A equipe de desenvolvimento detalha a arquitetura do sistema, a navegação, as interfaces de usuário e a modelagem do banco de dados, assegurando boa usabilidade e eficiência.

Entre as atividades desta fase, destaca-se a elaboração de diagramas de fluxo de dados, de entidade-relacionamento, de classes, protótipos de interface e diagramas arquiteturais. O objetivo é garantir que as estruturas projetadas sejam suficientes para suportar todas as funcionalidades do sistema. Também são identificadas dependências, pontos de integração e eventuais restrições, como limitações de *hardware* e requisitos de desempenho.

O resultado desta fase é o Documento de *Design* de Software (DDS), comumente denominado *Software Design Document* (SDD) na literatura, que estrutura formalmente as informações do projeto e aborda preocupações de *design*. Neste documento, são incluídos os artefatos produzidos, servindo como guia para coordenar equipes grandes e garantir que todos os componentes do sistema funcionem de maneira integrada.

## Codificação

Na fase de codificação, engenheiros e desenvolvedores transformam o *design* do *software* em código executável, com o objetivo de produzir um *software* funcional, eficiente e com boa usabilidade. Para isso, são utilizadas linguagens de programação adequadas, seguindo o DDS e as diretrizes de codificação estabelecidas pela organização e pela legislação local.

Durante essa fase, são realizadas revisões de código, nas quais os membros da equipe examinam o trabalho uns dos outros para identificar erros ou inconsistências, garantindo

elevados padrões de qualidade. Além disso, testes preliminares internos são conduzidos para assegurar que as funcionalidades básicas do sistema sejam atendidas.

Ao final da codificação, o *software* passa a existir como um produto funcional, representando a materialização dos esforços das etapas anteriores, mesmo que ainda sejam necessários refinamentos e ajustes subsequentes. O resultado desta fase é o código-fonte.

## Testes

A fase de testes consiste em verificar a qualidade e a confiabilidade do *software* antes de sua entrega aos usuários finais. O objetivo é identificar falhas, erros e vulnerabilidades, assegurando que o sistema atenda aos requisitos especificados.

Inicialmente, são definidos parâmetros de teste alinhados aos requisitos do *software* e casos de teste que contemplam diferentes cenários de uso. Em seguida, são realizados testes de diversos níveis e tipos, incluindo testes de unidade, integração, sistema, segurança e aceitação, permitindo a avaliação tanto de componentes individuais quanto da operação do sistema como um todo.

Quando um erro é identificado, ele é registrado detalhadamente, incluindo seu comportamento, métodos de reprodução e impacto sobre o sistema. As falhas são encaminhadas para correção e o *software* retorna à fase de testes para validação. Esse ciclo de teste e correção se repete até que o sistema esteja conforme os critérios previamente estabelecidos. O resultado desta fase é um código-fonte mais robusto e menos propenso a falhas.

## Implantação

A fase de implantação (*deployment*) consiste em disponibilizar o *software* aos usuários finais, garantindo sua operacionalidade no ambiente de produção. Esse processo ocorre tanto no primeiro lançamento do sistema quanto durante atualizações, devendo minimizar interrupções e impactos no acesso dos usuários.

Além de colocar o *software* em operação, esta fase envolve garantir que os usuários compreendam seu funcionamento. Para isso, podem ser fornecidos manuais, treinamentos e suporte técnico. Assim, a implantação marca a transição do *software* de projeto para produto, iniciando efetivamente o cumprimento de seus objetivos e a entrega de valor ao usuário.

## Manutenção

A fase de manutenção é caracterizada pelo suporte contínuo e por melhorias incrementais, garantindo que o *software* mantenha seu funcionamento adequado, acompanhe as necessidades dos usuários e as demandas do mercado. Nessa fase, são realizadas atualizações, correções de falhas e suporte ao usuário. Considerando o longo prazo, a manutenção inclui estratégias de modernização ou substituição do *software*, buscando manter sua relevância e adequação às evoluções tecnológicas.

### 1.1.2 Ferramentas de desenvolvimento

As ferramentas de desenvolvimento de *software* oferecem suporte às etapas do SDLC e evoluíram significativamente nas últimas décadas. Além das ferramentas tradicionais, observa-se a incorporação crescente de técnicas de inteligência artificial, em especial aprendizado de máquina e modelos de linguagem de grande escala, com o objetivo de aumentar a produtividade, reduzir erros e apoiar a tomada de decisão ao longo do processo de desenvolvimento (SCHMIDT *et al.*, 2024; HUANG *et al.*, 2024).

O uso combinado dessas ferramentas contribui para maior produtividade, qualidade e confiabilidade no desenvolvimento. Elas incluem:

- **Controle de Versão (como Git e SVN):** permitem registrar e gerenciar alterações no código-fonte ao longo do tempo, possibilitando colaboração simultânea entre desenvolvedores, recuperação de versões anteriores e rastreamento completo do histórico de mudanças;
- **Ambientes de Desenvolvimento Integrados (IDEs) (como Visual Studio, IntelliJ IDEA e Eclipse):** oferecem um conjunto de ferramentas em um único ambiente, incluindo edição de código, depuração, testes, gerenciamento de dependências, integração com sistemas de controle de versão e ferramentas de IA generativa, como assistentes de código;
- **Gerenciamento de Projetos (como Jira, Trello e Asana):** auxiliam na organização e priorização de tarefas, acompanhamento do progresso e comunicação entre membros da equipe, fornecendo transparência e facilitando a coordenação do trabalho;
- **Integração e Entrega Contínua (CI/CD) (como Jenkins, GitHub Actions e GitLab CI):** automatizam processos de compilação, testes e implantação, promovendo maior qualidade e agilidade nas entregas de *software*;
- **Teste (como Selenium, JUnit e Postman):** permitem a execução de testes automatizados e manuais para validar funcionalidades, desempenho e segurança do sistema, contribuindo para a detecção precoce de falhas e melhoria da qualidade do *software*.
- **Virtualização e Monitoramento (como Docker, Prometheus e Grafana):** permitem criar ambientes isolados e consistentes para execução do *software*, garantindo que ele funcione de maneira idêntica em diferentes máquinas. Além disso, possuem a capacidade de monitorar o desempenho e a saúde dos sistemas em produção, auxiliando na detecção precoce de problemas.

## 1.2 Inteligência artificial

Inteligência Artificial (IA) é o campo da ciência da computação dedicado à criação de sistemas capazes de executar tarefas que normalmente exigiriam inteligência humana, como reconhecimento de padrões, raciocínio, tomada de decisão, resolução de problemas e aprendizado a partir de dados. Segundo declaração da IEEE (*Artificial Intelligence* 2019), a inteligência artificial inclui tecnologias computacionais inspiradas no modo como pessoas

e outros organismos biológicos percebem, aprendem, raciocinam e agem.

As aplicações de IA afetam cada vez mais diversos aspectos da sociedade, incluindo defesa e segurança nacional, sistemas de justiça, comércio, finanças, manufatura, saúde, transporte, educação, entretenimento e interações sociais. Essas aplicações se expandem pela combinação de processadores avançados, grandes volumes de dados e novos algoritmos. Segundo estudo da [McKINSEY GLOBAL INSTITUTE \(2018\)](#), a IA contribuirá com cerca de 13 trilhões de dólares para o PIB global até 2030.

### **1.2.1 Aprendizado de máquina**

Aprendizado de Máquina (*Machine Learning*) é um subcampo da inteligência artificial que se concentra no desenvolvimento de algoritmos capazes de aprender padrões a partir de dados e realizar previsões ou tomadas de decisão sem serem explicitamente programados para cada tarefa específica ([MITCHELL, 1997](#)). Em geral, esses algoritmos utilizam grandes volumes de dados, frequentemente estruturados ou rotulados, para ajustar seus parâmetros e melhorar seu desempenho ao longo do tempo ([BISHOP, 2007](#)).

Um conceito central no aprendizado de máquina são as redes neurais artificiais, modelos computacionais inspirados na estrutura do cérebro humano, compostos por camadas de nós interconectados, que processam informações e aprendem padrões a partir de exemplos ([GOODFELLOW et al., 2016](#)). Redes neurais têm sido amplamente empregadas em tarefas como classificação, reconhecimento de imagens e processamento de linguagem natural, apresentando resultados expressivos em diferentes domínios da computação.

### **1.2.2 Aprendizado profundo**

Aprendizado Profundo (*Deep Learning*) é uma subárea do aprendizado de máquina que utiliza redes neurais artificiais com múltiplas camadas para aprender representações hierárquicas e abstratas dos dados. Esse tipo de abordagem permite capturar relações complexas em grandes volumes de informação, sendo fundamental para avanços recentes em visão computacional, reconhecimento de fala e processamento de linguagem natural ([GOODFELLOW et al., 2016](#)).

O progresso do aprendizado profundo está diretamente associado ao aumento da capacidade computacional, à disponibilidade de grandes volumes de dados e ao desenvolvimento de arquiteturas de redes neurais mais profundas e expressivas. Esses fatores foram determinantes para os avanços recentes em diversas áreas da inteligência artificial, especialmente em tarefas relacionadas ao processamento de linguagem natural ([LECUN et al., 2015](#)).

### **1.2.3 IA generativa**

A IA generativa é um ramo da inteligência artificial voltado à criação de novos conteúdos, como textos, imagens, códigos, áudios e vídeos, a partir de padrões aprendidos em grandes conjuntos de dados. Diferentemente de modelos que se concentram em classificação ou predição, modelos gerativos aprendem distribuições complexas e são capazes de produzir saídas originais e contextualmente coerentes ([BROWN et al., 2020](#)). No contexto da engenharia de *software*, a IA generativa tem sido aplicada em atividades como geração

de código, documentação automática, criação de casos de teste e apoio à revisão de código ([HUANG et al., 2024](#)).

#### **1.2.4 Modelos de linguagem de grande escala (LLMs)**

Entre as principais tecnologias associadas à IA generativa estão os Modelos de Linguagem de Grande Escala (*Large Language Models* ou LLMs). Esses modelos são baseados na arquitetura de *transformers*, proposta por [VASWANI et al. \(2017\)](#), que emprega mecanismos de atenção para capturar dependências de longo alcance em sequências, especialmente textuais, possibilitando a modelagem mais eficaz do contexto ao longo da geração de conteúdo.

Treinados com grandes volumes de textos, códigos-fonte e documentos, os LLMs demonstram capacidade de realizar uma ampla variedade de tarefas, incluindo sumarização, tradução, classificação, análise semântica, escrita de textos e geração de código. Modelos de linguagem de grande escala são capazes de aprender distribuições complexas da linguagem natural e gerar textos novos que preservam coerência semântica e contextual ao longo de sequências extensas, mesmo sem ajuste supervisionado específico para cada tarefa ([BROWN et al., 2020](#)).

# Capítulo 2

## Estratégia e seleção das fontes

Este capítulo descreve a abordagem metodológica adotada para a condução da pesquisa, detalhando os critérios utilizados para a seleção das fontes, a organização das análises e o método empregado para a comparação entre literatura formal e literatura cinzenta. O trabalho segue um processo analítico que parte de uma fonte central de literatura formal, a partir da qual outras fontes, de literatura cinzenta, foram selecionadas a fim de complementar, contrastar e aprofundar a análise.

### 2.1 Seleção das fontes

A seleção das fontes teve como objetivo garantir relevância temática, atualidade e diversidade de perspectivas. Dessa forma, foram consideradas tanto fontes acadêmicas quanto fontes oriundas do mercado, incluindo estudos empíricos, relatórios técnicos, pesquisas de opinião e análises baseadas em métricas de código. Essa combinação permitiu contemplar diferentes pontos de vista, como os de pesquisadores, desenvolvedores e executivos, bem como evidências quantitativas extraídas de bases de código reais.

Na engenharia de *software*, diferentemente de muitas outras áreas do conhecimento, conferências e congressos frequentemente possuem peso equivalente ou até superior ao de periódicos científicos tradicionais, em função da rápida evolução tecnológica da área. Por esse motivo, a seleção das fontes não se restringiu a artigos publicados em periódicos acadêmicos, mas incluiu também relatórios técnicos e pesquisas de mercado amplamente utilizados na indústria. As fontes foram organizadas em dois grupos: literatura formal e literatura cinzenta.

#### 2.1.1 Literatura formal

No grupo de literatura formal, o artigo de SERGEYUK *et al.* (2025) foi adotado como ponto de partida da pesquisa. Essa escolha se justifica por se tratar da revisão de literatura mais recente identificada no momento da seleção das fontes, publicada no primeiro semestre de 2025, além de apresentar ampla cobertura dos aspectos técnicos e práticos relacionados ao uso de assistentes de código baseados em inteligência artificial.

O artigo encontra-se indexado em bases reconhecidas, como o *Information and Software Technology* da Elsevier, um periódico relevante na área de engenharia de software. Trata-se de um estudo revisado por pares, conduzido por autores com forte vínculo com a indústria, majoritariamente pesquisadores da JetBrains, o que favorece uma abordagem alinhada a práticas reais de desenvolvimento. O artigo investiga como desenvolvedores utilizam assistentes de inteligência artificial, as razões para sua não adoção em determinadas etapas do SDLC e os aspectos que necessitam de aprimoramento, tornando-o uma referência adequada para estruturar e orientar a análise realizada neste trabalho.

### 2.1.2 Literatura cinzenta

O grupo de literatura cinzenta foi composto por relatórios e estudos que pudesse complementar a análise sob diferentes perspectivas empíricas e de mercado. O relatório de **GITCLEAR (2025)** concentra-se na análise de métricas de código extraídas de repositórios abertos e de empresas, construindo uma série histórica de resultados entre 2020 e 2024. A relevância dessa fonte decorre do fato de a GitClear atuar diretamente na análise automatizada de repositórios de código em larga escala, possuindo acesso contínuo a bases de código reais tanto de projetos de código aberto quanto de ambientes corporativos.

As pesquisas conduzidas pelo **STACK OVERFLOW (2023; 2024; 2025)** compilam tendências relacionadas ao perfil e aos hábitos de uso de ferramentas de IA por desenvolvedores usuários da plataforma. A relevância dessas fontes está associada à posição do *Stack Overflow* como uma das maiores comunidades globais de desenvolvedores, reunindo milhões de usuários ativos de diferentes níveis de experiência, setores e regiões. Assim, os dados coletados refletem percepções e práticas amplamente disseminadas na comunidade de desenvolvimento de software. Neste trabalho, essas pesquisas foram organizadas de forma comparativa, possibilitando a análise de mudanças temporais e a identificação de padrões recorrentes no uso de ferramentas de IA.

Os relatórios de **COSHOW et al. (2024)** e **BATCHU et al. (2024)**, publicados pela *Gartner Research*, oferecem uma perspectiva de mercado sobre o uso de inteligência artificial generativa no desenvolvimento de software, com foco em tendências emergentes, riscos percebidos e expectativas de executivos e gestores de tecnologia. A *Gartner* é reconhecida internacionalmente como uma das principais consultorias de pesquisa e análise estratégica em tecnologia da informação, sendo amplamente utilizada por organizações para embasar decisões de investimento e adoção tecnológica. Por esse motivo, seus relatórios fornecem uma visão consolidada das expectativas e preocupações do ponto de vista organizacional e gerencial.

Por fim, o relatório de **AI Coding Assistants (2024)** foi produzido em colaboração por duas agências governamentais europeias, a francesa, ANSSI, e a alemã, BSI, ambas responsáveis por diretrizes e políticas de segurança da informação em seus respectivos países. O documento reúne e sintetiza resultados de pesquisas acadêmicas relacionadas ao uso de assistentes de código baseados em inteligência artificial, com ênfase em riscos, limitações e implicações de segurança. Essa fonte ocupa uma posição intermediária entre literatura formal e cinzenta, sendo utilizada neste trabalho tanto como uma consolidação de evidências acadêmicas quanto como um arcabouço de referência para embasar e estruturar a análise comparativa.

## 2.2 Análises

O [Capítulo 3](#) foi dedicado à apresentação detalhada das fontes de literatura formal e cinzenta analisadas neste trabalho. Para cada fonte, investigou-se o contexto da organização responsável, o método empregado para a coleta e análise dos dados e os resultados mais relevantes para os objetivos da pesquisa. A [Seção 3.1](#) concentra-se na apresentação da fonte de literatura formal, enquanto a [Seção 3.2](#) aborda as fontes de literatura cinzenta.

Os conteúdos apresentados no [Capítulo 3](#) serviram de base para a análise desenvolvida no [Capítulo 4](#), cujo objetivo foi identificar consensos e divergências entre as diferentes fontes. Além disso, nesse capítulo, o relatório de [AI Coding Assistants \(2024\)](#) é utilizado de forma mais direta como um arcabouço de referência acadêmica, auxiliando na contextualização e no embasamento dos resultados discutidos.



# Capítulo 3

## Fontes

Nesse capítulo, estão apresentadas as fontes analisadas. Há seções dedicadas a cada fonte, com o resumo do seu conteúdo, seus resultados relevantes para este trabalho e um breve contexto da organização por trás dela. A divisão das fontes foi feita entre literatura formal e literatura cinzenta.

### 3.1 Literatura formal

A literatura formal é composta por trabalhos que passaram por um rigoroso processo de controle de qualidade antes da publicação. Os trabalhos desta literatura têm as seguintes características:

- **Revisão por pares (*peer review*):** são avaliados anonimamente por outros especialistas (*blind review*) para garantir validade técnica e metodologia correta;
- **Indexação:** são facilmente encontrados em bases de dados bibliográficas confiáveis (*IEEE Xplore, ACM Digital Library, Scopus, Elsevier*);
- **Publicação comercial/acadêmica:** são publicados por editoras ou organizações estabelecidas.

Como fonte de literatura formal foi utilizado o artigo de [SERGEYUK \*et al.\* \(2025\)](#). Sua apresentação está detalhada a seguir.

#### 3.1.1 “Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward” (2025)

Esta fonte é um artigo científico revisado por pares, publicado na *Information and Software Technology*, revista renomada na área de engenharia de software, pertencente à *Elsevier*. O artigo foi elaborado por autores com forte vínculo com a indústria, parte significativa deles é pesquisador da *JetBrains*. O objetivo do estudo foi compreender como os desenvolvedores utilizam assistentes de inteligência artificial, identificar os motivos pelos quais essas ferramentas não são adotadas em determinadas etapas do fluxo de desenvolvimento e apontar aspectos que demandam aprimoramento futuro.

Para isso, a pesquisa foi realizada com uma quantidade expressiva de pessoas, a coleta de dados incluiu entrevistas com 481 programadores de 71 países diferentes. A amostra da pesquisa é composta principalmente por programadores experientes, dos quais 48,9% possuem mais de 10 anos de experiência e 28,9% possuem mais de 15 anos. Do total de participantes, 74,4% estão empregados em tempo integral em empresas e organizações, enquanto outros atuam parcialmente, são autônomos ou *freelancers*. A maioria (86,7%) se identifica como desenvolvedor de *software*, mas a amostra também inclui outras posições técnicas relevantes, como engenheiros *DevOps*, arquitetos de *software* e líderes de equipe.

Em particular, foram estudadas as seguintes cinco grandes atividades de engenharia de *software*: implementação de novas funcionalidades, escrita de testes, triagem de erros, refatoração e produção de artefatos em linguagem natural, assim como suas etapas individuais. A pesquisa incluiu 38 perguntas principais, sendo 5 abertas, o que permitiu aos participantes expressarem livremente suas opiniões. A pesquisa obteve 547 respostas completas e, após uma filtragem cuidadosa, foram analisadas 481 respostas.

As principais contribuições dessa pesquisa são:

- **Padrões de uso:** A pesquisa revela que 84,2% dos participantes utilizam assistentes de IA ocasional ou regularmente, sendo a implementação de novas funcionalidades a atividade mais popular para o uso desses assistentes. Entre as etapas individuais, gerar e resumir código são as mais amplamente utilizadas;
- **Áreas de foco:** Considerando quais atividades os desenvolvedores consideram menos agradáveis e desejam delegar, bem como em quais etapas os assistentes de IA são utilizados, o artigo destaca as áreas nas quais a comunidade acadêmica precisa se concentrar para gerar valor aos usuários. Isso inclui a escrita de testes de forma geral e, em particular, a geração de dados e recursos para testes, assim como a escrita de artefatos em linguagem natural;
- **Razões para não usar IA:** Foram identificados 20 temas distintos que representam diferentes motivos pelos quais os desenvolvedores não utilizam assistentes de IA, com os principais motivos sendo: falta de necessidade de assistência de IA, saída gerada pela IA imprecisa, falta de confiança do usuário e desejo de manter controle, e compreensão insuficiente do contexto pelo assistente de IA. Os dados em detalhes estão na Tabela 3.1;
- **Áreas para melhorias futuras:** Com base na prevalência das diferentes razões, foram formuladas as principais áreas de trabalho futuro necessárias para superar as limitações apontadas pelos respondentes. Entre elas estão a melhoria dos sistemas base, melhor integração no fluxo de trabalho dos desenvolvedores e educação maisativa dos usuários sobre as capacidades e limitações dos assistentes.

O gráfico da Figura 3.1 mostra as ferramentas que os respondentes utilizam em seu trabalho. De modo geral, 84,2% dos participantes mencionaram utilizar ao menos uma ferramenta. Entre as ferramentas mais citadas, destacam-se *ChatGPT* (72,1%), *GitHub Copilot* (37,9%), *JetBrains AI Assistant* (28,9%) e *Visual Studio IntelliCode* (17,5%). Quanto às demais ferramentas, mais da metade dos respondentes não as conhece. Isso indica que o campo de assistentes de IA é dominado por algumas grandes companhias.

Motivo	Resp. (%)
Falta de necessidade de assistência de IA	22.5
Saída gerada pela IA imprecisa	17.7
Falta de confiança do usuário e desejo de ter controle	15.7
Falta de compreensão do contexto pelo assistente de IA	14.4
Conhecimento limitado do usuário sobre assistentes de IA e suas capacidades	10.2
Desejo do usuário de realizar a tarefa sozinho e aprender	7.3
Ineficiência de tempo	5.5
Saída gerada pela IA não é útil	4.6
Políticas da empresa, NDA, etc.	3.6
Atitude negativa do usuário em relação à IA	3.2
Falta de conformidade com requisitos não funcionais na saída do assistente de IA	3.2
Considerações legais e éticas	3.1
Desafios de integração e usabilidade da IA em ambientes de desenvolvimento	3.0
Falta de acesso	2.6
Preocupações com segurança	2.3
Limitações na criatividade da IA	2.2
Interrupção do fluxo de trabalho	2.1
Limitações de conhecimento do modelo de IA	1.9
Ineficiência para tarefas específicas	1.7
Desafios na comunicação das intenções do usuário ao assistente de IA	1.0

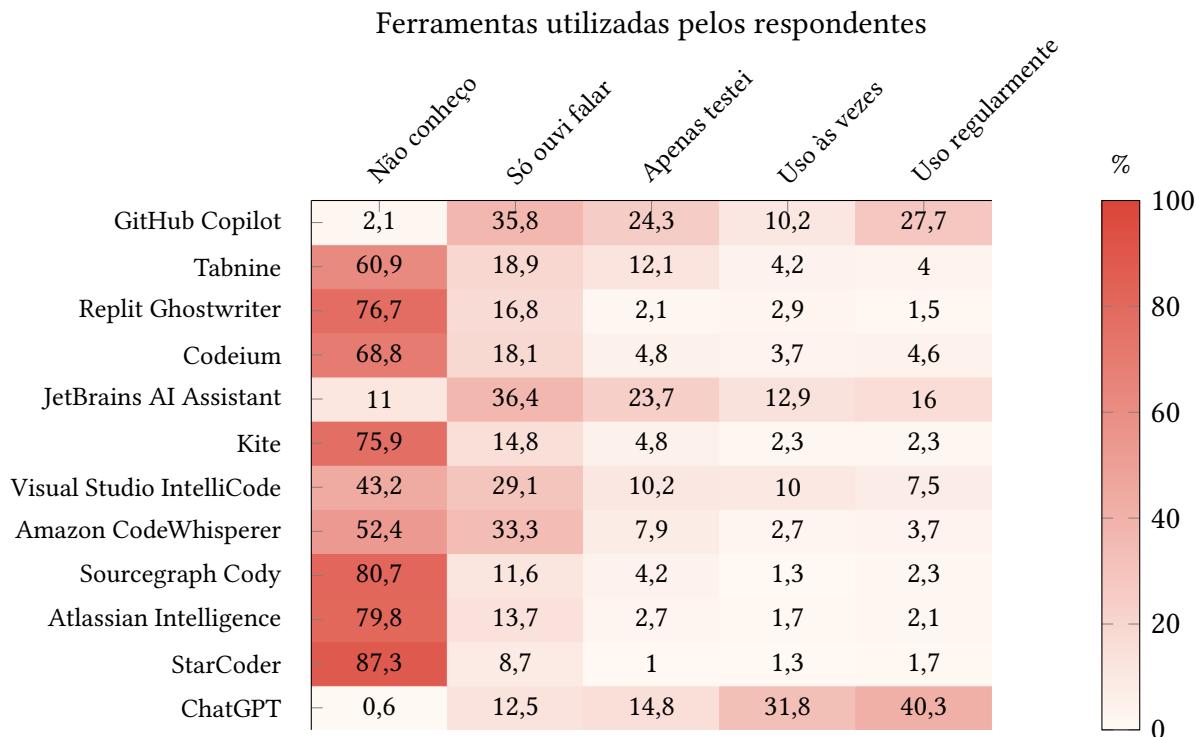
**Tabela 3.1:** Temas resultantes nas respostas à pergunta "O que impede você de usar assistentes de IA nas etapas do fluxo de trabalho que você não os utiliza IA?"(SERGEYUK *et al.*, 2025).

Como mostrado na [Figura 3.2](#), implementar novas funcionalidades é a atividade mais agradável para os participantes, com até 86% dos respondentes avaliando-a positivamente. Também é a atividade menos provável de ser delegada a um assistente de IA, com 48% dos participantes respondendo negativamente. Por outro lado, escrever testes e produzir artefatos em linguagem natural são as atividades menos agradáveis (46% de avaliações negativas para testes e 43% para artefatos), sendo também as que os desenvolvedores mais desejam delegar, 70% e 66%, respectivamente.

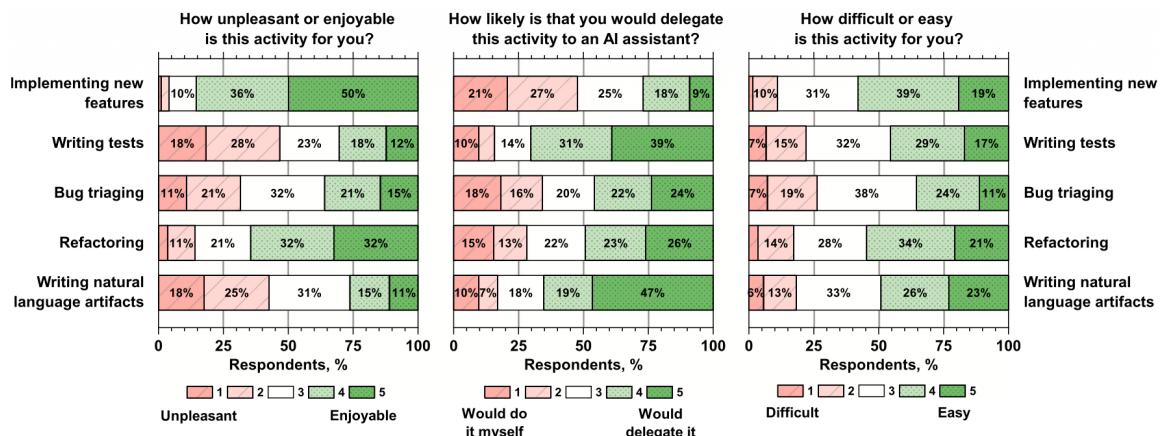
Refatoração é a segunda atividade mais agradável, com 64% de avaliações positivas. Mesmo assim, 49% dos participantes considerariam delegar essa atividade a um assistente de IA. Quanto à triagem de erros, as opiniões são mais divididas: 32% dos respondentes a consideram desagradável e 36% agradável, enquanto 34% não a delegariam e 46% estariam dispostos a delegá-la a uma IA.

O artigo organiza seus principais resultados em quatro *takeaways*, eles são:

- A opinião geral sobre o código fornecido pelos assistentes de IA varia, sendo o aspecto mais positivo a sua usabilidade e o mais negativo a segurança;
- Entre as principais atividades estudadas, implementar novas funcionalidades é a mais



**Figura 3.1:** Ferramentas utilizadas pelos desenvolvedores (SERGEYUK et al., 2025; n = 481)



**Figura 3.2:** Gráfico representando quais tarefas do desenvolvimento os desenvolvedores gostam de fazer, quais eles delegariam à IA e qual sua impressão de dificuldade sobre as atividades (SERGEYUK et al., 2025).

agradável e a menos propensa a ser delegada a um assistente pelos respondentes, enquanto escrever testes e produzir artefatos em linguagem natural são as mais desagradáveis e as mais propensas a serem delegadas;

- As funcionalidades mais utilizadas de assistentes de IA estão associadas a geração e resumo, por exemplo, de códigos, testes, refatorações, artefatos de linguagem natural e correções de erros. Por outro lado, usar a IA para modificar autonomamente o repositório ou utilizá-la para identificar pontos onde novas funcionalidades, testes e refatorações devem ser feitos são usos menos populares, apesar de não serem

negligenciáveis entre os respondentes;

- Os principais motivos pelos quais os desenvolvedores não usam assistentes de IA são: falta de necessidade, saída gerada pela IA imprecisa, falta de confiança e falta de compreensão do contexto pelo assistente.

## 3.2 Literatura cinzenta

A literatura cinzenta refere-se a materiais de pesquisa produzidos por governos, academia, empresas e indústria, em formatos impressos ou eletrônicos, que não passam pelos canais tradicionais de publicação comercial. Em outras palavras, trata-se de conhecimento que não foi submetido ao funil tradicional das editoras acadêmicas.

Na engenharia de *software*, a literatura cinza é valiosa. Exemplos incluem:

- **Relatórios técnicos (*Technical Reports*)**: documentos internos de empresas que detalham novas arquiteturas;
- **White papers**: documentos informativos emitidos por empresas para promover ou destacar uma solução ou tecnologia;
- **Teses e dissertações**: embora acadêmicas, muitas vezes são consideradas cinzas até que o conteúdo seja publicado em artigo;
- **Normas e padrões**: documentos da ISO ou do W3C;
- **Blogs de engenharia e documentação**: posts em blogs oficiais de engenharia ou documentação de APIs.

As apresentações das fontes de literatura cinzenta analisadas nesse trabalho estão detalhadas a seguir.

### 3.2.1 *AI Copilot Code Quality: Evaluating 2024's Increased Defect Rate via Code Quality Metrics (2025)*

Este relatório foi escrito pela empresa GitClear, que desenvolve ferramentas de análise de código-fonte para empresas de desenvolvimento de *software*. O relatório realizou uma análise de métricas de qualidade de código de 211 milhões de linhas alteradas em 2024, sendo dois terços destas linhas advindos do compartilhamento de dados anonimizados de empresas privadas e o restante de projetos de código livre. Os pesquisadores criaram um histórico comparando as métricas observadas em pesquisas anteriores com as métricas de 2024 e a análise é feita com foco no impacto de ferramentas de IA generativa no processo de desenvolvimento de *software*, considerando 2022 como o ano em que a adoção da IA na programação se iniciou.

Na **Tabela 3.2**, são apresentados os resultados das análises entre 2020 e 2024. Observa-se que a porcentagem de linhas movidas, associada à refatoração, algo essencial para reduzir a dívida técnica e garantir a manutenibilidade do sistema a longo prazo, apresentou uma queda significativa, passando de 24,1% em 2020 para apenas 9,5% em 2024.

Ano	Adicionado	Deletado	Atualizado	Movido	Cópia	Substituído	Churn
2020	39,2%	19,1%	5,2%	24,1%	8,3%	2,9%	3,1%
2021	39,5%	19,3%	5,0%	24,8%	8,4%	3,4%	3,3%
2022	40,9%	19,8%	5,2%	20,5%	9,4%	3,7%	3,3%
2023	42,3%	21,1%	5,6%	15,8%	10,6%	3,6%	4,5%
2024	46,2%	21,9%	5,9%	9,5%	12,3%	4,2%	5,7%

**Tabela 3.2:** Histórico de métricas de código de 2020 a 2024 ([GITCLEAR, 2025](#)).

Ano	Tot. commits	Tot. duplicações	Commits com duplicação	Commits com duplicação (%)
2020	19.805	9.227	139	0,70%
2021	29.912	9.295	143	0,48%
2022	40.010	10.685	182	0,45%
2023	41.561	20.448	747	1,80%
2024	56.495	63.566	3.764	6,66%

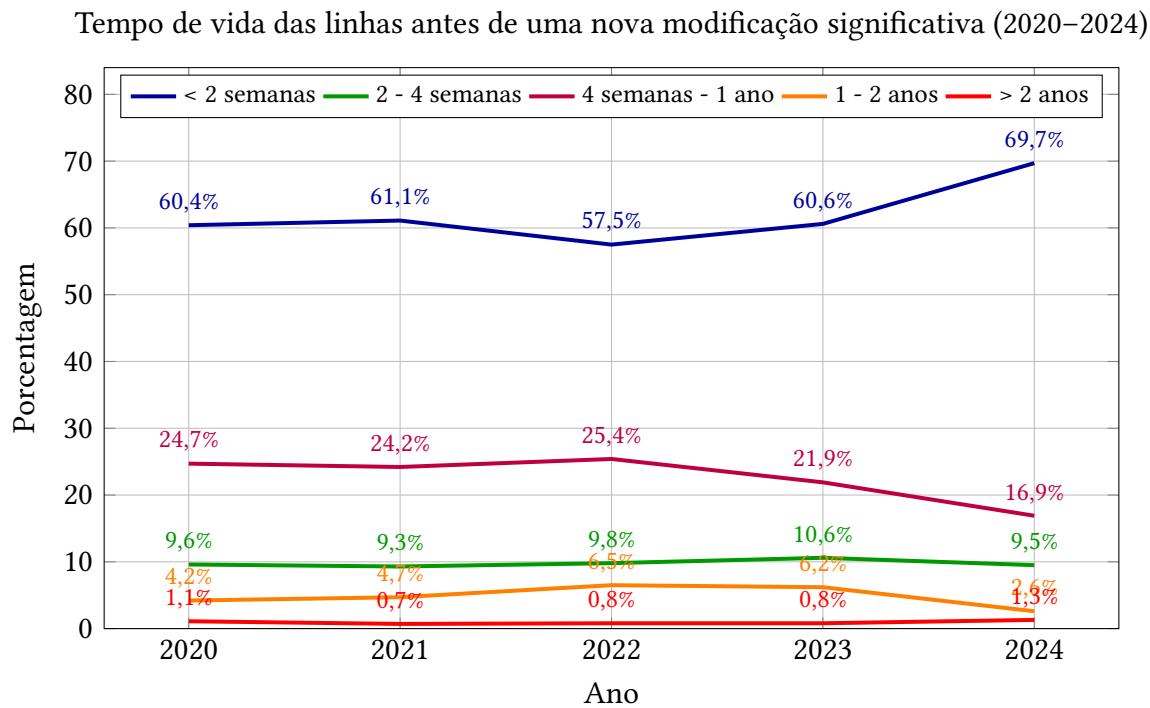
**Tabela 3.3:** Histórico de duplicação de código nos commits de 2020 a 2024 ([GITCLEAR, 2025](#)).

Por outro lado, a tendência para as linhas copiadas foi inversa, partindo de 8,3% em 2020 para 12,3% em 2024. Este aumento de duplicações também pode ser observado na [Tabela 3.3](#), que mostra que, em 2020, apenas 0,7% dos commits analisados continham blocos de código duplicados e, em 2024, esse número subiu para 6,66%. A duplicação de código, além de violar o princípio DRY (*Don't Repeat Yourself*), segundo [HUMMEL et al. \(2009\)](#), aumenta a probabilidade de erros no código, já que as cópias tendem a evoluir de forma inconsistente.

Observa-se também um aumento de 2,9% para 4,2% entre 2020 e 2024 na proporção de linhas substituídas. O relatório credita essa mudança ao uso de IDEs e assistentes de código de IA, que podem com facilidade reescrever trechos de código para conformar-se às regras de linting e às convenções do projeto.

Além disso, outra métrica relevante neste contexto é o *churn*, que busca medir quanto retrabalho é efetuado em uma base de código. Neste relatório, o *churn* é calculado medindo quanto das linhas escritas e enviadas ao repositório foram revertidas ou substancialmente revisadas nas duas semanas seguintes. A [Tabela 3.2](#) mostra que o *churn* passou de 3,1% em 2020 para 5,7% em 2024, evidenciando o aumento do retrabalho no código.

O relatório fez também uma análise de quanto tempo se passou entre o momento em que os códigos analisados foram criados e em quanto tempo eles receberam sua próxima mudança significativa. Na [Figura 3.3](#), observa-se que a proporção de linhas que em menos de duas semanas foram alteradas aumentou de 60,4% em 2020 para 69,7% em 2024, em contrapartida, o grupo das linhas que precisaram de alterações apenas depois de 4 semanas, mas antes de um ano, diminuiu de 24,7% para 16,9% no mesmo período.



**Figura 3.3:** Proporção anual das linhas modificadas classificadas pelo tempo que permaneceram no repositório antes de sofrerem uma nova modificação significativa entre 2020 e 2024 (GITCLEAR, 2025).

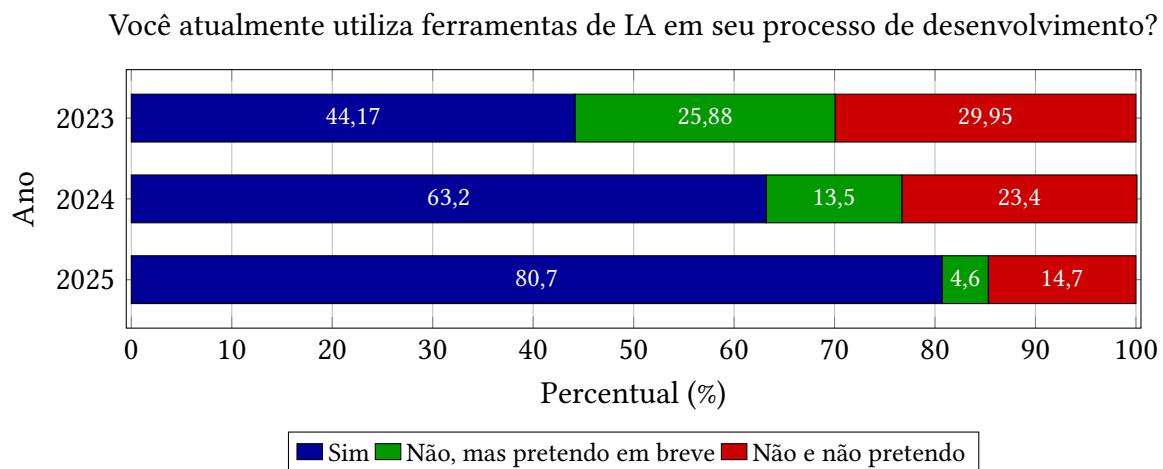
### 3.2.2 Stack Overflow Developer Survey (2023; 2024; 2025)

Esta seção compila os resultados de 3 anos da *Stack Overflow Developer Survey* de 2023 a 2025. Esta pesquisa é realizada anualmente pela *Stack Overflow*, uma das principais plataformas *online* de perguntas e respostas voltadas para desenvolvedores de *software*, que possibilita que programadores de todo o mundo tirem dúvidas técnicas, compartilhem soluções e aprendam a partir de problemas reais.

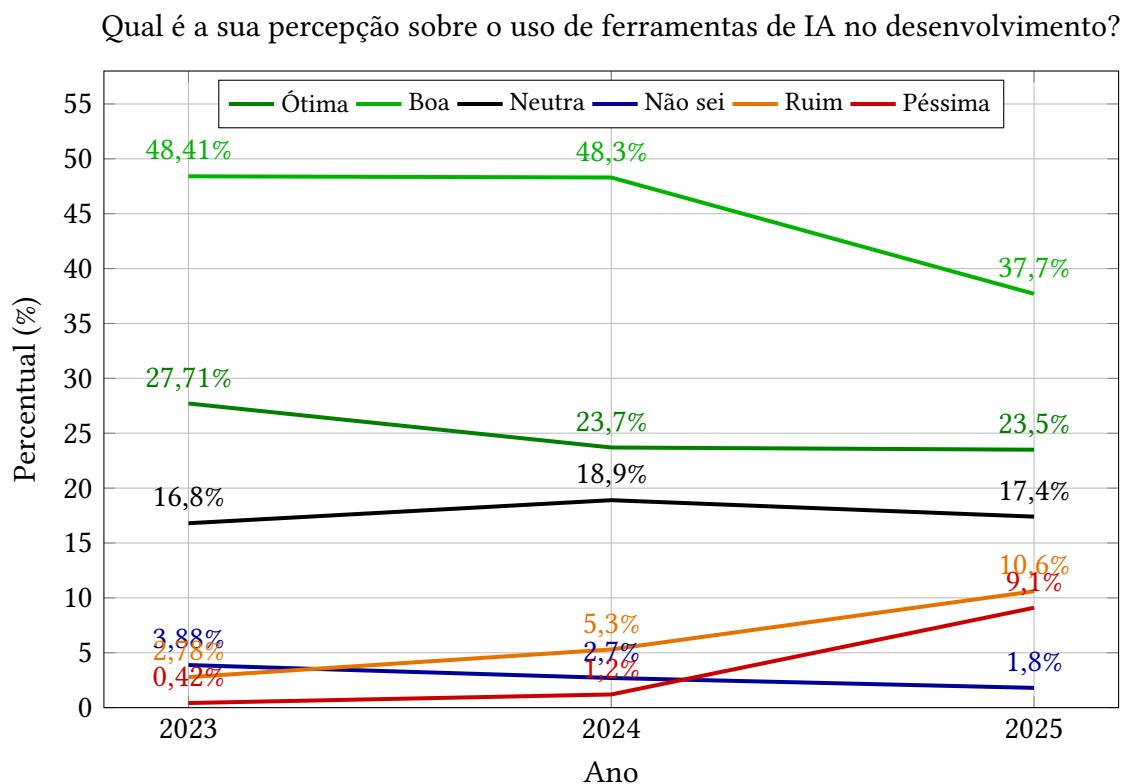
A pesquisa é aberta, incluindo respondentes de todo o mundo, desde programadores profissionais até pessoas que estão iniciando seus estudos sobre programação. A pesquisa coleta dados sobre o uso de tecnologias, linguagens de programação, ferramentas, práticas de desenvolvimento, perfil profissional e, mais recentemente, uso de IA no processo de desenvolvimento de *software*. Os resultados das perguntas geralmente permitem fazer o recorte dos respondentes quanto ao seu conhecimento de programação, as respostas dadas por programadores profissionais foram priorizadas nesta seção.

Na **Figura 3.4**, estão apresentados os dados referentes à adoção de ferramentas de IA por parte dos desenvolvedores profissionais que participaram das pesquisas ao longo dos três anos. O gráfico captura o forte crescimento na adoção destas ferramentas, partindo de 44,17% de uso em 2023 para 80,7% em 2025.

Na **Figura 3.5**, estão apresentados os dados referentes à percepção dos desenvolvedores sobre uso de IA no desenvolvimento de *software*. As impressões dos programadores têm mudado de pesquisa para pesquisa. De 2023 a 2025, as porcentagens de profissionais que têm uma percepção boa ou ótima sobre as ferramentas de IA caíram de 48,41% para 37,7% e de 27,71% para 23,5%, respectivamente. Por outro lado, no mesmo período, as percepções



**Figura 3.4:** Proporção anual dos desenvolvedores profissionais que utilizam ferramentas de IA em seu processo de desenvolvimento entre 2023 e 2025 (STACK OVERFLOW, 2023; 2024; 2025;  $n_{2023} = 67237$ ;  $n_{2024} = 46049$ ;  $n_{2025} = 33662$ ).

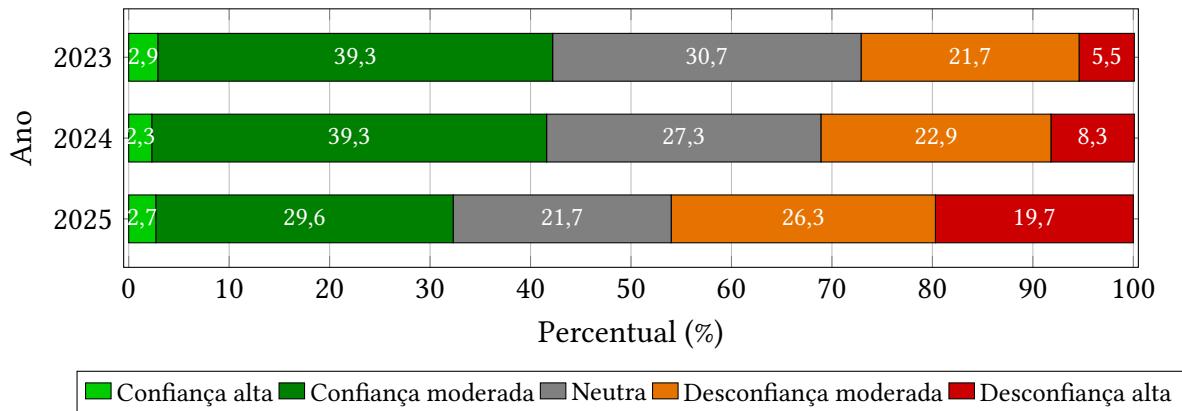


**Figura 3.5:** Proporção anual da percepção dos desenvolvedores profissionais em relação às ferramentas de IA de 2023 a 2025 (STACK OVERFLOW, 2023; 2024; 2025;  $n_{2023} = 46928$ ;  $n_{2024} = 35142$ ;  $n_{2025} = 25814$ ).

ruim e péssima subiram de 2,78% para 10,6% e de 0,42% para 9,1%, respectivamente. A mudança fica ainda mais evidente ao agrupar as percepções positivas e negativas, já que as positivas caíram de 76,12% para 61,2% e as negativas subiram de 3,2% para 19,7%, um crescimento bastante expressivo.

Seguindo a tendência de piora, a Figura 3.6 apresenta o grau de confiança dos pro-

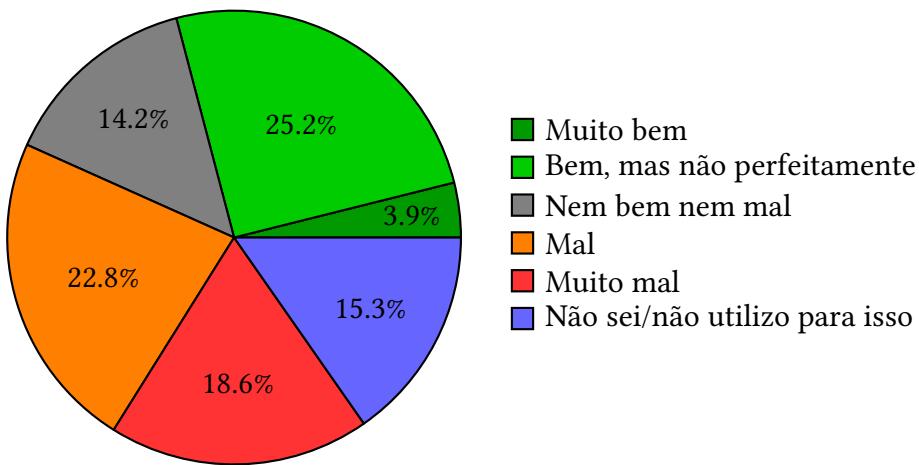
Qual é o seu nível de confiança no uso de ferramentas de IA no desenvolvimento?



**Figura 3.6:** Proporção anual do nível de confiança dos desenvolvedores em ferramentas de IA entre 2023 e 2025 (STACK OVERFLOW, 2023; 2024; 2025;  $n_{2023} = 39042$ ;  $n_{2024} = 28829$ ;  $n_{2025} = 25701$ ).

gramadores em relação aos resultados apresentados pelas ferramentas de IA. De 2023 a 2025, as proporções de profissionais com confiança moderada ou alta em ferramentas de IA caíram de 39,3% para 29,6% e de 2,85% para 2,7%, respectivamente. Em contrapartida, no mesmo período, as desconfianças moderada e alta subiram de 21,71% para 26,3% e de 5,46% para 19,7%, respectivamente. Como no dado anterior, agrupar os graus positivos e negativos evidencia a queda, já que a confiança foi de 42,15% para 32,3%, enquanto a desconfiança subiu de 27,17% para 46%, ou seja, em 2025, menos respondentes confiam nas ferramentas de IA que o contrário.

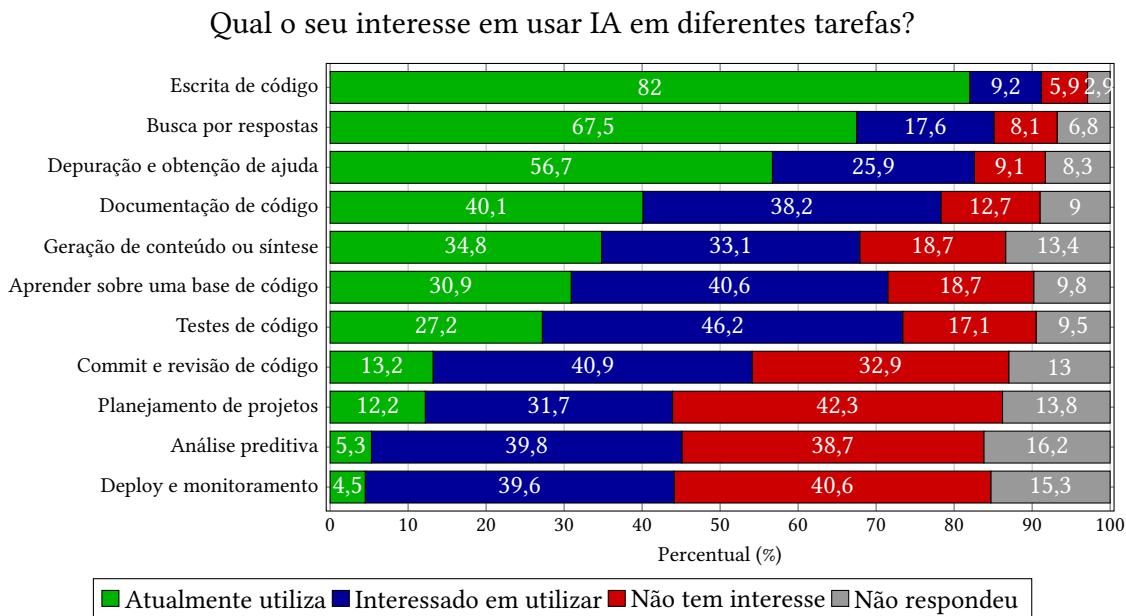
Quão bem as ferramentas de IA que você usa em seu fluxo de trabalho de desenvolvimento lidam com tarefas complexas?



**Figura 3.7:** Percepção dos desenvolvedores profissionais sobre a capacidade das ferramentas de IA em lidar com tarefas complexas em 2025 (STACK OVERFLOW, 2025;  $n = 25695$ ).

A Figura 3.7 mostra a percepção dos desenvolvedores profissionais em 2025 em relação à capacidade das ferramentas de IA em lidar com tarefas complexas. Embora cerca de 29,1% dos respondentes avaliem o desempenho como positivo, uma parcela significativa considera

o desempenho como insatisfatório, com cerca de 41% classificando-a como mal ou muito mal. Esses resultados indicam que, apesar dos avanços, ainda há limitações relevantes no uso de IA para atividades de maior complexidade no desenvolvimento de *software*. Este gráfico trouxe apenas os dados de 2025 porque a pergunta não foi feita em 2023 e em 2024, as alternativas para resposta mudaram, dificultando a comparação entre anos.



**Figura 3.8:** Uso atual e interesse futuro no uso de ferramentas de IA ao longo do fluxo de desenvolvimento em 2024 (STACK OVERFLOW, 2024; n = 35978)

A Figura 3.8 apresenta os principais usos que os programadores têm feito das ferramentas de IA em 2024. O gráfico mostra que as atividades para as quais os desenvolvedores mais estão utilizando IA são: escrita de código (82%), busca por respostas (67,5%) e depuração e obtenção de ajuda (56,7%). Por outro lado, as tarefas para as quais os desenvolvedores estão menos interessados em utilizar IA são: planejamento de projetos (42,3%), implantação e monitoramento (40,6%) e análise preditiva (38,7%). Este gráfico inclui apenas os dados de 2024, pois, em 2023, a pergunta permitia respostas de modo que as categorias de uso e interesse se sobreponham e, em 2025, havia mais categorias possíveis, incluindo "usa majoritariamente IA", "parcialmente IA", entre outros, dificultando a comparação com anos anteriores.

Apesar disso, a categoria que cobre o fato dos respondentes não terem interesse em usar IA para uma determinada tarefa está presente nas três pesquisas. Na Tabela 3.4, está compilado o histórico dessa categoria de 2023 a 2025 e a tendência para todas as atividades foi de diminuição do interesse no uso de IA. Deploy e monitoramento e planejamento de projetos foram as tarefas com as maiores porcentagens, 75,8% e 69,2%, ou seja, a maioria dos programadores não tem interesse em usar IA nelas.

### 3.2.3 Magic Quadrant for AI Code Assistants (2024)

Este relatório foi produzido por pesquisadores da *Gartner Research*. A *Gartner* é uma empresa de pesquisa e consultoria especializada em tecnologia da informação e gestão

Atividade	2023 (%)	2024 (%)	2025 (%)
Deploy e monitoramento	28,3	40,6	75,8
Planejamento de projetos	29,8	42,3	69,2
Análise preditiva	-	38,7	65,6
Commit e revisão de código	23,0	32,9	58,7
Testes de código	11,4	17,1	44,1
Aprender sobre uma base de código	13,1	18,7	39,4
Documentação de código	8,1	12,7	38,5
Geração de conteúdo ou dados sintéticos	-	18,7	38,2
Depuração ou correção de código	6,4	9,1	36,4
Escrita de código	4,5	5,9	28,9
Busca por respostas	-	8,1	19,6

**Tabela 3.4:** Proporção de desenvolvedores que não têm interesse em utilizar ferramentas de IA em diferentes partes do SDLC entre 2023 e 2025. As atividades presentes em apenas uma das pesquisas não foram incluídas (*STACK OVERFLOW, 2023; 2024; 2025; n<sub>2023</sub> = 37726; n<sub>2024</sub> = 35978; n<sub>2025</sub> = 25349*).

empresarial, fundada em 1979. A organização é reconhecida pela produção de relatórios analíticos, previsões de mercado e estudos prospectivos sobre tendências tecnológicas, amplamente utilizados por empresas e organizações como suporte à tomada de decisão estratégica.

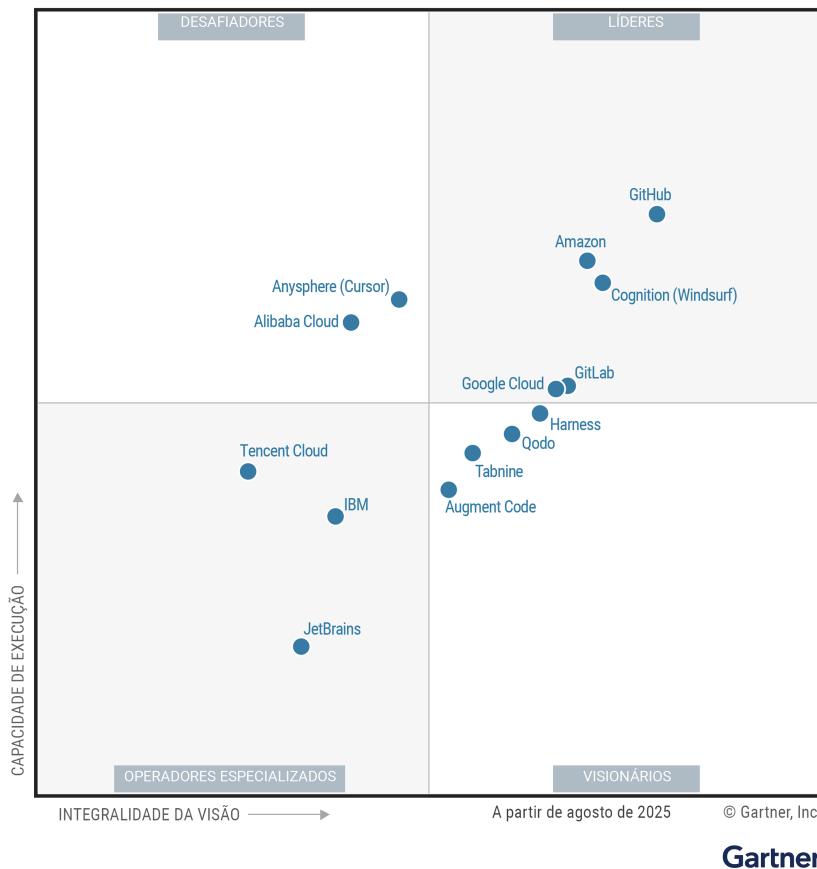
Neste relatório, há uma análise de diferentes aspectos relativos aos assistentes de código baseados em IA. As previsões do relatório sobre essas ferramentas são:

- De 2024 até 2027, o número de equipes de engenharia de plataforma que utilizarão IA para aprimorar todas as fases do SDLC aumentará de 5% para 40%;
- Até 2027, 80% das empresas integrarão ferramentas de teste com IA em sua cadeia de produção de *software*, representando um aumento significativo em relação a cerca de 15% no início de 2023;
- Até 2027, 25% dos defeitos de *software* que chegarão ao ambiente de produção resultarão da falta de supervisão humana sobre código gerado por IA, um aumento expressivo em comparação com menos de 1% em 2023;
- Até 2028, 90% dos engenheiros de *software* em ambientes corporativos utilizarão assistentes de código baseados em IA, frente a menos de 14% no início de 2024;
- Até 2028, o uso de IA generativa reduzirá em 30% os custos de modernização de aplicações legadas em relação aos níveis de 2023.

As previsões do relatório apontam para a continuidade da tendência de adoção de IA generativa na engenharia de *software*. Além disso, o texto destaca vantagens dessas ferramentas, como poder aprimorar a experiência do desenvolvedor de *software* ao impulsionar o desenvolvimento de aplicações, minimizar a sobrecarga cognitiva dos programadores, ampliar suas habilidades de resolução de problemas, acelerar o ritmo de aprendizado e fomentar a criatividade.

Outro ponto levantado pelo relatório são as funcionalidades essenciais para que essas ferramentas atendam às necessidades de mercado, as funcionalidades são:

- Completar código a partir de linguagem natural, como comentários.
- Completar código multilinha, podendo gerar código no meio de um arquivo de forma coerente com o que veio antes e depois;
- Capacidade de funcionar em diferentes editores de texto, IDEs, ambientes e plataformas de desenvolvimento;
- Garantia de que os modelos não serão treinados com o conteúdo dos repositórios do cliente (exceto quando permitido).
- Interface de *chat* conversacional integrada ao ambiente de desenvolvimento.



**Figura 3.9:** Quadrante Mágico dos assistentes de código com IA (BATCHU et al., 2024)

Na Figura 3.9, a *Gartner* classificou as empresas que atuam no mercado de assistentes de código de IA num gráfico formado pelos eixos, “capacidade de execução” e “integralidade da visão”. No gráfico, é evidente a posição de destaque do *GitHub* com o *GitHub Copilot*, seu assistente de código lançado como prévia técnica em junho de 2021 e disponível para o público em geral no *Visual Studio Code* a partir de junho de 2022.

Segundo o relatório, o *GitHub Copilot* tem como foco aumentar a produtividade dos desenvolvedores e a qualidade do código através de sugestões de código baseadas em IA e

assistência contextual. Suas operações são geograficamente diversificadas e seus clientes tendem a ser grandes organizações de diversos setores.

Uma estratégia destacada pelo relatório é o fato do *GitHub* disponibilizar o *GitHub Copilot Pro* gratuitamente para mantenedores ativos da comunidade de *software* livre, professores e estudantes, fidelizando novos programadores desde seus estudos na faculdade. Além disso, o *Copilot* teve novas funcionalidades adicionadas como o *Copilot Workspace*, que oferece suporte a um ambiente colaborativo de desenvolvimento nativo em IA, e as *Copilot Extensions*, apoiadas por um ecossistema crescente de parceiros de *software* livre e comerciais, que permitem que desenvolvedores construam soluções integradas a seus ambientes de desenvolvimento preferidos.

O relatório atribui o sucesso da ferramenta, entre outros fatores, ao fato do *GitHub* ser uma subsidiária da *Microsoft*, se beneficiando de uma ampla comunidade global de desenvolvedores. Desta maneira, a empresa de consultoria acredita que o *GitHub Copilot* apresenta alto engajamento de usuários, o que permite à empresa coletar *feedbacks* em larga escala, garantindo a melhora da ferramenta de forma contínua.

De modo geral, neste relatório, a visão da *Gartner* quanto ao mercado dos assistentes de código parece bastante positiva, tanto no sentido da continuidade da adoção destas ferramentas quanto em relação a sua capacidade de promover a melhora da produtividade nas empresas. Além disso, fica visível a posição de destaque do *GitHub Copilot* neste mercado.

### **3.2.4 Top Strategic Technology Trends for 2025: Agentic AI (2024)**

Este relatório também foi produzido por pesquisadores da *Gartner Research*. Nele, o foco é apresentar oportunidades, riscos, previsões e recomendações quanto ao uso de IA agêntica em empresas para executivos que contratam os serviços de consultoria da *Gartner*. Além disso, também é trazida a perspectiva de CIOs sobre os resultados esperados com o uso de IA generativa.

O termo IA agêntica se refere a entidades de *software* que receberam direitos da organização para tomar decisões e agir em seu nome de forma autônoma. Diferentemente da automação robótica de processos, a IA agêntica não requer entradas explícitas e não produz saídas predeterminadas.

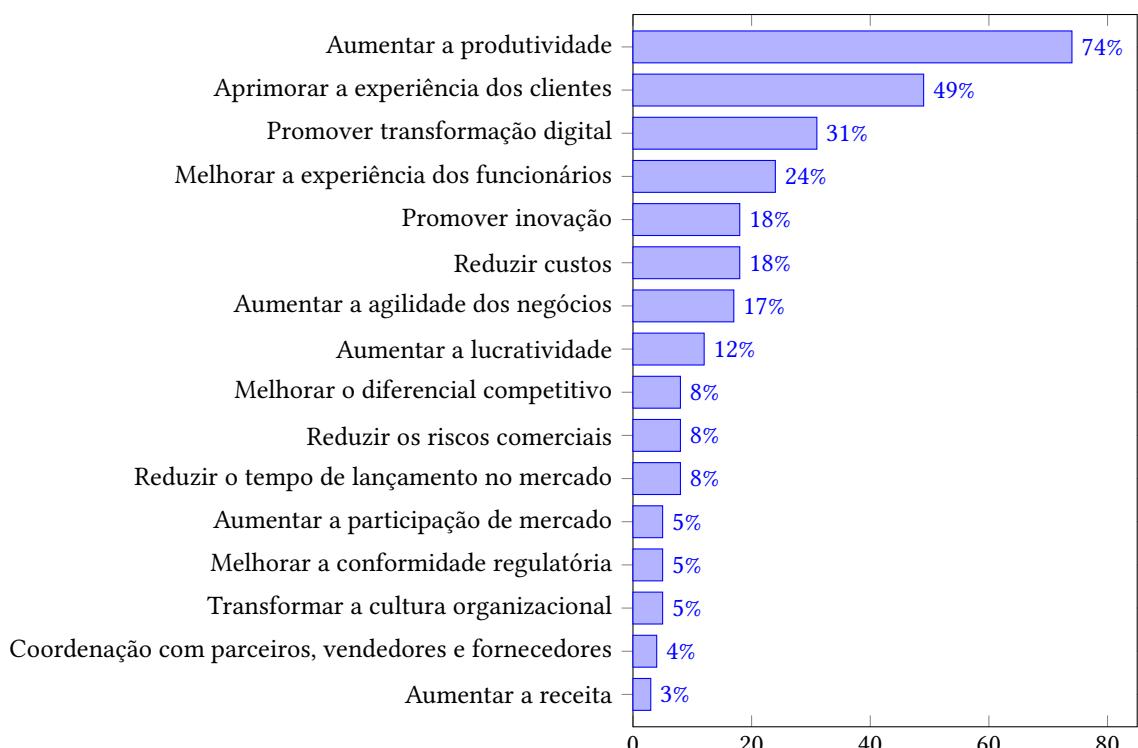
Segundo o relatório, a IA agêntica será “uma extensão da força de trabalho que não necessita de férias ou de outros benefícios” e as previsões da empresa indicam que ela será capaz de:

- Aumentar a produtividade nas empresas de forma expressiva;
- Capacitar trabalhadores, permitindo que eles desenvolvam e gerenciem projetos técnicos complexos através de linguagem natural;
- Mudar os processos decisórios nas empresas, já que conseguirá analisar conjuntos de dados complexos, identificando padrões e agindo. Isso evitaria modelagens de dados trabalhosas, aperfeiçoará a resolução de problemas, reduzirá o tempo de ação e permitirá análises em maior escala.

Por outro lado, a empresa de consultoria destaca que a orquestração e a governança dessas entidades autônomas de *software* requerem ferramentas avançadas e proteções rigorosas. Por isso, a *Gartner* definiu recomendações de uso de IA agêntica e elas são:

- Estabelecer diretrizes legais e éticas sobre autonomia, responsabilidade, segurança e privacidade de dados, garantindo que os recursos sejam robustos o suficiente para proteger a organização, funcionários e clientes;
- Implementar proteções para assegurar que a IA agêntica se restrinja a uma função e um conjunto de recursos definidos, evitando que ela adote medidas incorretas ou prejudiciais;
- Projetar soluções de IA agêntica para conectar diferentes aplicativos e dados, priorizando a experiência e a eficiência dos usuários;
- Repensar fluxos de trabalho de áreas com demanda por escala e eficiência, automatizando o que for viável e adicionando pessoas de volta em pontos estratégicos;
- Começar em pequena escala com casos de uso com dados de qualidade disponíveis;
- Tratar agentes de IA como colegas digitais de nível básico, equivalentes a posições júnior, ou como funcionários digitais aos quais se delega trabalho;
- Dedicar a IA agêntica a tarefas como descobrir e fornecer percepções de eventos derivados que pessoas talvez não notem;

#### Resultados esperados por CIOs quanto ao uso de IA generativa em suas empresas



**Figura 3.10:** Principais resultados esperados por CIOs nos negócios com a aplicação da IA generativa (*COSHOW et al., 2024*)

O relatório destaca que o ganho de desempenho dessas ferramentas aumentará gradativamente à medida que os sistemas evoluírem, deste modo, o número e o uso de agentes de IA aumentarão para atender às demandas de produtividade. O texto evidencia o risco de que as organizações criem milhares de bots, sem que ninguém se lembre do que esses bots fazem ou porque foram criados, e que os funcionários implantem suas próprias IAs sem atender aos padrões de segurança ou qualidade das empresas.

Além disso, outra questão trazida no artigo é que a IA agêntica tomará decisões com base na análise dos dados da própria organização, o que poderá ser perigoso, pois estes dados podem ser de baixa qualidade. Este risco, além de piorar a qualidade dos resultados da IA, também pode inibir o seu desenvolvimento.

Quanto às previsões de adoção, a *Gartner* indica uma forte tendência de adoção da IA agêntica nos próximos anos. Segundo o relatório, até 2028, 33% dos aplicativos de *software* empresariais incluirão IA agêntica, em comparação com menos de 1% em 2024 e que, até 2028, pelo menos 15% das decisões cotidianas no trabalho serão tomadas de forma autônoma por IAs agênticas, em comparação com 0% em 2024.

Na [Figura 3.10](#), o relatório compilou os resultados de uma pesquisa de opinião realizada com 78 CIOs de empresas, os quais foram perguntados "Quais são os três principais tipos de valor de negócios que sua empresa busca com a aplicação de IA generativa?". Com este gráfico e com os pontos levantados ao longo do relatório fica claro que o investimento das empresas em ferramentas de IA, em especial a agêntica, está fundamentalmente atrelado à crença de que a IA aumentará a produtividade dos funcionários. Ao mesmo tempo que o texto parece otimista em relação à adoção de IAs, ele destaca que esse movimento possui diversos riscos, os quais são vistos como responsabilidade das empresas usuárias mitigar.

### **3.2.5 *AI Coding Assistants (2024)***

O relatório foi elaborado conjuntamente pela *Agence Nationale de la Sécurité des Systèmes d'Information* (ANSSI), agência francesa responsável pela cibersegurança nacional, e pelo *Bundesamt für Sicherheit in der Informationstechnik* (BSI), o Escritório Federal Alemão para Segurança da Informação. Ambas são autoridades governamentais que atuam na definição de diretrizes, recomendações e boas práticas para a proteção de sistemas de informação e infraestrutura digital em seus respectivos países.

A publicação analisa as oportunidades e os riscos associados ao uso de assistentes de programação baseados em IA, tecnologia que já é amplamente adotada em organizações e tende a se tornar parte integrante do desenvolvimento de *software*. O relatório começa com uma introdução conceitual sobre o que são assistentes de código com IA e o objetivo do documento. Em seguida, o texto é dividido em seções temáticas que discutem, separadamente, as oportunidades associadas ao uso dessas ferramentas, os riscos de segurança envolvidos e, por fim, um conjunto de recomendações práticas. A conclusão consolida essas recomendações, segmentando-as por níveis organizacionais (gestão, desenvolvimento e pesquisa), o que confere ao texto um caráter normativo e orientado à aplicação prática, típico de publicações institucionais de órgãos de segurança da informação.

O conteúdo em detalhes do artigo não foi compilado na íntegra neste trabalho, pois é bastante denso e repleto de citações diversas. Em vez disso, seu conteúdo e suas referências

serão utilizados diretamente na análise presente no [Capítulo 4](#).

# Capítulo 4

## Análise

Neste capítulo, com base em todos os dados levantados, foi realizada uma análise que reuniu os diferentes trabalhos e cruzou as informações mais importantes entre si. Os artigos e relatórios foram escolhidos porque abordam diferentes aspectos do tema, desde um ponto de vista da indústria até abordagens acadêmicas.

### 4.1 Amadurecimento da percepção dos programadores

A mudança no sentimento dos desenvolvedores a respeito da IA ficou bastante visível na compilação dos resultados das pesquisas do *Stack Overflow* (2023; 2024; 2025). Apesar de o uso de ferramentas de IA continuar crescendo ([Figura 3.4](#)), tanto a percepção dos desenvolvedores sobre a qualidade das respostas das ferramentas de IA para a engenharia de software ([Figura 3.5](#)) quanto a confiança dos profissionais em seus conteúdos gerados ([Figura 3.6](#)) pioraram significativamente nos últimos anos.

Isso talvez explique, o aumento na proporção de tarefas as quais os desenvolvedores se apresentam desinteressados a adotar ([Tabela 3.4](#)). Ano após ano, os programadores apresentaram mais resistência a usar IA para atividades de alta responsabilidade ou de complexidade sistêmica como *deploy* e monitoramento ou planejamento de projetos ([Tabela 3.4](#)).

### 4.2 Indícios da adoção e extração

Este trabalho reuniu diversos relatórios que estimam o uso de ferramentas de IA por parte dos desenvolvedores. Dado o tamanho das amostras destas pesquisas, não é possível afirmar com relevância estatística que de fato todos os programadores do mundo estão adotando estas ferramentas de IA nestas proporções, no entanto, há alguns indícios interessantes.

A *Stack Overflow* (2023; 2024; 2025) tem apresentado consistentemente um aumento nas declarações de programadores que utilizam IA ([Figura 3.4](#)). Outra evidência, também relacionada ao *Stack Overflow*, foi documentada por [RIO-CHANONA et al. \(2024\)](#), que

analisou a forte queda nos acessos à plataforma, relacionando-a com o aumento do uso de ferramentas de IA para tirar dúvidas de código ou fazer triagem de erros.

Além disso, a própria Gartner, através do relatório escrito por [COSHOW et al. \(2024\)](#) e do escrito por [BATCHU et al. \(2024\)](#), evidencia suas previsões de aumento tanto do uso de IA generativa em assistentes de código quanto através de ferramentas de IA agêntica. A pesquisa de [SERGEYUK et al. \(2025\)](#) também corrobora com essa visão de aumento, já que 84,2% dos participantes da pesquisa declarou utilizar assistentes de IA.

### 4.3 Desconexão entre discurso e prática

Como evidenciado no artigo de [SERGEYUK et al. \(2025\)](#), os programadores entrevistados se declararam mais propensos a delegar tarefas que gostam menos para as ferramentas de IA ([Figura 3.2](#)), no entanto, a própria pesquisa evidencia que o maior uso das ferramentas de IA entre os participantes é justamente a produção de código, a tarefa apontada na pesquisa como a mais proveitosa pelos programadores e a que eles menos delegariam a uma IA ([Figura 3.2](#)). Este é um ponto interessante a se avaliar em pesquisas futuras, no sentido do impacto dessa ação para a satisfação profissional da categoria.

### 4.4 Tendência de automatização

Como evidenciado pelo artigo de [COSHOW et al. \(2024\)](#), da *Gartner*, o mercado não apenas caminha para a continuidade do uso de ferramentas de IA nos diversos setores empresariais, incluindo a produção de *software*, mas também para o crescimento da adoção da IA agêntica. No contexto de empresas de *software*, a IA agêntica não apenas sugere e gera código, como também manipula a base de código diretamente.

Segundo [ZIEGLER et al. \(2022\)](#), os programadores tendem a se sentir mais produtivos conforme aceitam mais sugestões de seus assistentes de código. Essa informação somada à previsão da *Gartner* apresentada por [BATCHU et al. \(2024\)](#) de que 25% dos erros de produção serão advindos de código de IA não revisado, já indicam a necessidade da criação de mecanismo que impeçam os programadores de adicionar linhas não revisadas às bases de código. Isto será especialmente desafiador dado o crescimento esperado da IA agêntica.

### 4.5 As métricas e o futuro

A queda na qualidade das métricas evidenciado pela [GITCLEAR \(2025\)](#) apresenta não apenas um cenário negativo em relação ao aumento da duplicação de código, quanto da queda da refatoração. Neste contexto, é relevante investigar como essa piora da qualidade impactará a sustentabilidade das empresas e de que maneira essas métricas evoluirão, conforme a adoção de IA avançar.

# Capítulo 5

## Conclusão

O trabalho conclui, portanto, que o cenário provável para o futuro é de que a tendência de adoção de IA continue se expandindo pelas empresas. Os indícios até o momento sugerem que a qualidade dos códigos escritos na “era da IA” tem se deteriorado e que um dos principais usos feitos pelos programadores de ferramentas de IA é justamente a escrita de código.

De alguma maneira, os programadores parecem estar reagindo a essa piora na qualidade do código. A percepção dos profissionais sobre as ferramentas de IA tem ano após ano se convertendo numa visão mais sóbria e menos positiva. Essa mudança tem impactado os programadores nos seus usos de IA, de modo que tarefas de alta complexidade e responsabilidade tem sido menos confiadas à IA.

Este trabalho se baseou bastante em literaturas cinzentas para a sua análise e é importante que novas pesquisas sejam feitas no sentido de formalizar muito do que já aparece na literatura informal. Algumas comparações de pesquisas foram difíceis, inclusive porque não há um padrão muito bem definido até mesmo entre pesquisas de uma mesma empresa.

Outras análises relevantes seriam possíveis no sentido de continuar o acompanhamento das fontes apresentadas, já que muitas delas são anuais e poderão atualizar o parecer atual sobre percepções, impactos, métricas de código, entre outros. Além disso, com a adoção da IA agêntica em crescimento, é possível que novos impactos surjam, intensificando tendências já presentes em 2025. Este é outro foco relevante para a continuidade do tema.



# Referências

- [AI Coding Assistants 2024] *AI Coding Assistants*. Rel. técn. Security recommendations regarding AI programming assistants. France e Germany: Agence nationale de la sécurité des systèmes d'information (ANSSI) e Federal Office for Information Security (BSI), 2024. URL: [https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/KI/ANSSI\\_BSI\\_AI\\_Coding\\_Assistants.pdf](https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/KI/ANSSI_BSI_AI_Coding_Assistants.pdf) (citado nas pgs. 10, 11, 27).
- [Artificial Intelligence 2019] *Artificial Intelligence*. Position Statement. Piscataway, NJ, USA: IEEE, jun. de 2019. URL: <https://globalpolicy.ieee.org/wp-content/uploads/2019/06/IEEE18029.pdf> (citado na pg. 6).
- [BATCHU *et al.* 2024] Arun BATCHU, Philip WALSH, Matt BRASIER e Haritha KHANDA-BATTU. *Magic Quadrant for AI Code Assistants*. Rel. técn. Document 5682355. Gartner Research, 2024. URL: <https://www.gartner.com/en/documents/5682355> (citado nas pgs. 10, 22, 24, 30).
- [BISHOP 2007] Christopher M. BISHOP. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1<sup>a</sup> ed. Springer, 2007. ISBN: 0387310738. URL: <http://www.amazon.com/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0387310738> (citado na pg. 7).
- [BROWN *et al.* 2020] Tom B BROWN *et al.* “Language models are few-shot learners”. *arXiv preprint arXiv:2005.14165* (2020) (citado nas pgs. 7, 8).
- [COSHOW *et al.* 2024] Tom COSHOW *et al.* *Top Strategic Technology Trends for 2025: Agentic AI*. Rel. técn. ID G00818765. Acesso em: 9 dez. 2025. Gartner Research, out. de 2024. URL: [https://cdn.prod.website-files.com/6115505d46eace49d6ae6aa2/6762a0b88f83cb2cd933c5f7\\_818765.pdf](https://cdn.prod.website-files.com/6115505d46eace49d6ae6aa2/6762a0b88f83cb2cd933c5f7_818765.pdf) (citado nas pgs. 10, 25, 26, 30).
- [GITCLEAR 2025] GITCLEAR. *AI Copilot Code Quality: Evaluating 2024's Increased Defect Rate via Code Quality Metrics*. Rel. técn. Acessado em: 17 nov. 2025. GitClear, 2025. URL: <https://gitclear-public.s3.us-west-2.amazonaws.com/AI-Copilot-Code-Quality-2025.pdf> (citado nas pgs. 10, 17–19, 30).

- [GOODFELLOW *et al.* 2016] Ian GOODFELLOW, Yoshua BENGIO e Aaron COURVILLE. *Deep Learning*. Book in preparation for MIT Press. MIT Press, 2016. URL: <http://www.deeplearningbook.org> (citado na pg. 7).
- [HUANG *et al.* 2024] Yuan HUANG *et al.* *Generative Software Engineering*. 2024. arXiv: 2403.02583 [cs.SE]. URL: <https://arxiv.org/abs/2403.02583> (citado nas pgs. 6, 8).
- [HUMMEL *et al.* 2009] Benjamin HUMMEL, Elmar JUERGENS, Stefan WAGNER e Florian DEISSENBOECK. “Do code clones matter?” In: *2009 31st International Conference on Software Engineering (ICSE 2009)*. Los Alamitos, CA, USA: IEEE Computer Society, mai. de 2009, pp. 485–495. doi: 10.1109/ICSE.2009.5070547. URL: <https://doi.ieee.org/10.1109/ICSE.2009.5070547> (citado na pg. 18).
- [IEEE Standard Glossary of Software Engineering Terminology 1990] IEEE Standard Glossary of Software Engineering Terminology. Rel. técn. 1990, pp. 1–84. doi: 10.1109/IEEESTD.1990.101064 (citado na pg. 3).
- [JOHNSON e MENZIES 2024] Brittany JOHNSON e Tim MENZIES. “AI Over-Hype: A Dangerous Threat (and How to Fix It)”. *IEEE Software* 41.06 (nov. de 2024), pp. 131–138. ISSN: 1937-4194. doi: 10.1109/MS.2024.3439138. URL: <https://doi.ieee.org/10.1109/MS.2024.3439138> (citado na pg. 1).
- [LECUN *et al.* 2015] Yann LECUN, Yoshua BENGIO e Geoffrey HINTON. “Deep learning”. *nature* 521.7553 (2015), p. 436 (citado na pg. 7).
- [McKINSEY GLOBAL INSTITUTE 2018] MCKINSEY GLOBAL INSTITUTE. *Notes from the AI frontier: Modeling the impact of AI on the world economy*. Acesso em: 25 ago. 2025. Set. de 2018. URL: <https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-modeling-the-impact-of-ai-on-the-world-economy> (citado na pg. 7).
- [MITCHELL 1997] Tom M MITCHELL. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997 (citado na pg. 7).
- [RIO-CHANONA *et al.* 2024] R Maria del RIO-CHANONA, Nadzeya LAURENTSYEVA e Johannes WACHS. “Large language models reduce public knowledge sharing on online q&a platforms”. *PNAS Nexus* 3.9 (set. de 2024). Ed. por Matjaz PERC. ISSN: 2752-6542. doi: 10.1093/pnasnexus/pgae400. URL: <http://dx.doi.org/10.1093/pnasnexus/pgae400> (citado na pg. 29).
- [SCHMIDT *et al.* 2024] Douglas C. SCHMIDT, John Robert OZKAYA e Ipek OZKAYA. *Transforming Software Engineering and Software Acquisition with Large Language Models*. Software Engineering Institute, William & Mary. Capítulo técnico sobre uso de LLMs no SDLC. 2024. URL: <https://www.cs.wm.edu/~dcschmidt/PDF/LLM-chapter-AI-SDLC.pdf> (citado na pg. 6).

## REFERÊNCIAS

- [SERGEYUK *et al.* 2025] Agnia SERGEYUK, Yaroslav GOLUBEV, Timofey BRYKSN e Iftekhar AHMED. “Using ai-based coding assistants in practice: state of affairs, perceptions, and ways forward”. *Information and Software Technology* 178 (2025), p. 107610. ISSN: 0950-5849. DOI: <https://doi.org/10.1016/j.infsof.2024.107610>. URL: <https://www.sciencedirect.com/science/article/pii/S0950584924002155> (citado nas pgs. 9, 13, 15, 16, 30).
- [STACK OVERFLOW 2023] STACK OVERFLOW. *Stack Overflow Developer Survey*. 2023. URL: <https://survey.stackoverflow.co/2023/> (acesso em 19/08/2025) (citado nas pgs. 10, 19–21, 23, 29).
- [STACK OVERFLOW 2024] STACK OVERFLOW. *Stack Overflow Developer Survey*. 2024. URL: <https://survey.stackoverflow.co/2024/> (acesso em 19/08/2025) (citado nas pgs. 10, 19–23, 29).
- [STACK OVERFLOW 2025] STACK OVERFLOW. *Stack Overflow Developer Survey*. 2025. URL: <https://survey.stackoverflow.co/2025/> (acesso em 19/08/2025) (citado nas pgs. 1, 10, 19–21, 23, 29).
- [TERRAGNI *et al.* 2025] Valerio TERRAGNI, Annie VELLA, Partha ROOP e Kelly BLINCOE. “The future of ai-driven software engineering”. *ACM Trans. Softw. Eng. Methodol.* 34.5 (mai. de 2025). ISSN: 1049-331X. DOI: [10.1145/3715003](https://doi.org/10.1145/3715003). URL: <https://doi.org/10.1145/3715003> (citado na pg. 1).
- [VASWANI *et al.* 2017] Ashish VASWANI *et al.* “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008. URL: <http://arxiv.org/abs/1706.03762> (citado na pg. 8).
- [ZIEGLER *et al.* 2022] Albert ZIEGLER *et al.* “Productivity assessment of neural code completion”. In: *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*. MAPS 2022. San Diego, CA, USA: Association for Computing Machinery, 2022, pp. 21–29. ISBN: 9781450392730. DOI: [10.1145/3520312.3534864](https://doi.org/10.1145/3520312.3534864). URL: <https://doi.org/10.1145/3520312.3534864> (citado na pg. 30).

