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: Prof. Dr. Paulo Roberto Miranda Meirelles
Cossupervisor: Arthur Pilone Maia da Silva

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.

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, em forma de texto. Deve apresentar os objetivos, métodos empregados, resultados e conclusões. O resumo deve ser redigido em parágrafo único, conter no máximo 500 palavras e ser seguido dos termos representativos do conteúdo do trabalho (palavras-chave). Deve ser precedido da referência do documento. Texto texto. Texto texto.

Palavras-chave: Palavra-chave1. Palavra-chave2. Palavra-chave3.

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.

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa. De acordo com o Regimento da Pós-Graduação da USP (Artigo 99), deve ser redigido em inglês para fins de divulgação. É uma boa ideia usar o sítio www.grammarly.com na preparação de textos em inglês. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.

Keywords: Keyword1. Keyword2. Keyword3.

Lista de abreviaturas

ANSSI	*Agence Nationale de la Sécurité des Systèmes d'Information* (Agência Francesa de Segurança dos Sistemas de Informação)
BSI	Bundesamt für Sicherheit in der Informationstechnik (Escritório Alemão de Segurança da Informação)
CIO	*Chief Information Officer* (Diretor de Tecnologia da Informação)
IA	Inteligência Artificial
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
IME	Instituto de Matemática e Estatística
LLM	*Large Language Model* (Modelo de Linguagem de Grande Escala)
OTAN	Organização do Tratado do Atlântico Norte
SDLC	*Software Development Life Cycle* (Ciclo de Desenvolvimento de Software)
USP	Universidade de São Paulo

Lista de figuras

Lista de tabelas

Sumário

Introdução

Contexto

A engenharia de *software* é um campo da computação que se propõe a produzir e manter sistemas de *software*. Essa definição foi estabelecida em 1968 pela Organização do Tratado do Atlântico Norte (OTAN), direcionando esforços para resolver a chamada "crise do *software*", um período em que o desenvolvimento de programas se tornava cada vez mais complexo e desorganizado. Desde então, diversas ferramentas, métodos e processos foram criados para possibilitar que programadores organizassem a produção de *software* e realizassem projetos complexos com maior eficiência.

No mesmo período, a humanidade presenciou diversos avanços tecnológicos, como a produção de processadores cada vez mais potentes, o barateamento do *hardware*, tornando computadores e celulares muito mais acessíveis, e a inclusão de bilhões de pessoas na *internet*, gerando uma enorme quantidade de dados sobre os diversos aspectos da vida cotidiana e virtual. Com todo esse poder computacional e a quantidade massiva de dados disponíveis, a inteligência artificial (IA) pôde se desenvolver a passos largos, culminando no surgimento da IA generativa. Diferentemente da IA tradicional, a IA generativa é capaz de "criar" conteúdos com base no que aprendeu.

Dada sua flexibilidade, a IA generativa pode ser utilizada para diversos fins, e era natural que uma de suas aplicações fosse a engenharia de *software*. Nos últimos anos, vários estudos foram publicados com o objetivo de analisar essas aplicações, discutir suas consequências e propor abordagens seguras e responsáveis para seu uso (Johnson e Menzies, 2024 e Terragni *et al.*, 2025).

Segundo dados da Stack Overflow (2025), 80,7% dos desenvolvedores profissionais que participaram da pesquisa utilizam ferramentas de IA em seu processo de desenvolvimento de *software* e 4,6% deste grupo planeja utilizá-las em breve. É evidente que uma tecnologia amplamente adotada entre desenvolvedores tende a causar impactos significativos na produção de código e, nesse contexto, este trabalho faz-se relevante.

Objetivos

- Reunir dados sobre os impactos das ferramentas de IA generativa na produção de *software*;
- Analisar a evolução da percepção dos programadores sobre o uso dessas ferramentas;

- Comparar os resultados provenientes da literatura cinzenta e da literatura formal.

Estrutura do trabalho

O Capítulo 1 é uma fundamentação teórica, apresentando os conceitos essenciais para o trabalho. O Capítulo 2 descreve a metodologia adotada e as fontes utilizadas. No Capítulo 3, é apresentada uma revisão da literatura formal sobre o tema, já no Capítulo 4, a revisão foca na literatura cinzenta. O Capítulo 5 traz uma análise que compara e relaciona os estudos das duas frentes. Por fim, o Capítulo 6 conclui o trabalho.

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 (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), 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), 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 contemplem 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. 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;
- **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, possibilitam acompanhar 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 IA que se concentra na criação de algoritmos capazes de aprender e fazer previsões a partir de grandes volumes de

dados, geralmente estruturados ou rotulados, sem serem explicitamente programados para cada tarefa. Um conceito importante para o aprendizado de máquina são as redes neurais, modelos computacionais inspirados na estrutura do cérebro humano, compostos por camadas de nós interconectados, capazes de processar informações e aprender padrões a partir de exemplos. Redes neurais são amplamente utilizadas em tarefas como classificação, reconhecimento de imagens e processamento de linguagem natural.

1.2.2 Aprendizado Profundo

Aprendizado Profundo (*Deep Learning*) é uma área do aprendizado de máquina que utiliza redes neurais com múltiplas camadas para aprender representações cada vez mais abstratas dos dados. Conforme o número de camadas aumenta, essas redes se tornam capazes de extrair padrões complexos e hierárquicos, permitindo avanços significativos em tarefas cognitivas tradicionalmente difíceis para sistemas computacionais.

O avanço do aprendizado profundo está diretamente relacionado à disponibilidade de grandes volumes de dados, ao aumento da capacidade computacional e ao desenvolvimento de novas técnicas. Graças ao aprendizado profundo, problemas antes considerados inviáveis passaram a ter soluções com desempenho igual ou superior ao humano em diversos contextos.

1.2.3 IA Generativa

A IA generativa é um ramo da inteligência artificial focado na criação de novos conteúdos, como textos, imagens, códigos, áudios ou vídeos, a partir de padrões aprendidos em grandes conjuntos de dados. Diferentemente dos modelos tradicionais, que apenas classificam ou predizem valores específicos, os modelos generativos aprendem distribuições complexas e conseguem produzir saídas originais e coerentes com o contexto. Essa capacidade permitiu o desenvolvimento de aplicações como sistemas de criação de imagens, ferramentas de escrita automatizada, assistentes virtuais avançados e modelos de geração de código.

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 utilizam a arquitetura de *transformers*, baseada em mecanismos de atenção para capturar relações de longo alcance entre elementos de uma sequência. Treinados com bilhões de palavras, códigos e documentos, os LLMs podem realizar uma grande variedade de tarefas, incluindo resumo, tradução, classificação, análise semântica, escrita de textos e geração de código.

Capítulo 2

Metodologia

Este capítulo descreve a abordagem metodológica adotada para a condução da pesquisa, destacando os critérios utilizados para seleção das fontes, a forma de organização das análises e o método empregado para comparar literatura cinzenta e literatura formal. O trabalho segue um processo de análise partindo do artigo de literatura formal escrito por Sergeyuk *et al.* (2025). A partir deste artigo, outras fontes, principalmente de literatura cinzenta, foram selecionadas de modo a complementar a análise.

2.1 Seleção das fontes

A seleção das fontes foi feita buscando garantir a relevância, atualidade e diversidade de perspectivas. Desta forma, as fontes incluem perspectivas acadêmicas e de mercado, impressões de programadores e executivos e análises de métricas de código. As fontes foram divididas no grupo de literatura formal e literatura cinzenta.

2.1.1 Literatura formal

Neste grupo, o artigo de Sergeyuk *et al.* (2025), indexado em bases reconhecidas, como a *Elsevier*, foi utilizado como ponto de partida do trabalho. Também parte deste grupo, o artigo *AI Coding Assistants* (2024) foi produzido por duas agências governamentais europeias, a francesa, ANSSI, e a alemã, BSI, e é um compilado de resultados de pesquisas acadêmicas referentes ao uso de assistentes de código com IA.

2.1.2 Literatura cinzenta

Neste grupo, foram trazidos relatórios que pudessem complementar a análise. O artigo da GitClear (2025) foca na análise de métricas de código em repositórios abertos e de empresas, criando um histórico dos resultados de 2020 a 2024. As pesquisas do Stack Overflow (2023; 2024; 2025) compilam tendências no perfil de uso de IA dos usuários da plataforma e, neste trabalho, foram organizadas de modo a criar um comparativo entre os anos. Os artigos de Coshow *et al.* (2024) e Batchu *et al.* (2024) da *Gartner Research* trazem

uma visão de mercado sobre o uso de IA generativa no desenvolvimento de software, focando em tendências, riscos e expectativas de executivos.

2.2 Análises

No Capítulo 3, o foco é apresentar o texto de Sergeyuk *et al.* (2025), o ponto de partida do trabalho. O artigo aborda principais ferramentas de IA utilizadas pelos participantes da pesquisa, suas tendências de uso e impressões sobre o desempenho das ferramentas. Além disso, um breve contexto sobre o *AI Coding Assistants* (2024) é apresentado sem aprofundar tanto no conteúdo, já que ele é bastante denso.

Já o Capítulo 4 dedica-se às fontes de literatura cinzenta. Nele estão apresentados um resumo dos relatórios, seus resultados relevantes para este trabalho e um breve contexto da organização por trás de cada fonte.

Por fim, os conteúdos dos dois capítulos foram utilizados na análise realizada no Capítulo 5, cujo objetivo foi evidenciar consensos e divergências entre as fontes. Além disso, neste capítulo, o artigo *AI Coding Assistants* (2024) é utilizado de forma mais completa como arcabouço de fontes acadêmicas, ajudando a complementar e embasar a análise.

Capítulo 3

Literatura formal

3.1 "Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward" (2025)

In particular, we studied the following five broad SE activities: (a) implementing new features, (b) writing tests, (c) bug triaging, (d) refactoring, and (e) writing natural-language artifacts, as well as their individual stages. Our survey included 38 main questions, of them 5 being open-ended, which allowed the participants to freely express their thoughts. The survey attracted 547 complete responses, and after careful filtering, we study 481 responses. Our sample is experienced and diverse, with almost half of the respondents having more than 10 years of professional experience covering all the major programming languages and types of developed software. Overall, the contributions of this work are the following: • Usage patterns. We find that 84.2ally or regularly use AI assistants, that Implementing new features is the most popular activity to use assistants, and that among individual stages, generating and summarizing code are the most widely used. • Areas of focus. Taking into account which activities the develop- ers find less enjoyable and want to delegate, as well as what stages AI assistants are used on, we highlight areas where the research needs to focus on to bring value to users right now. This includes Writing tests in general and generating data and resources for tests in particular, as well as Writing natural language artifacts. • Reasons for not using AI. We provide 20 distinct themes that represent different reasons why developers are not using AI assis- tants and report the main ones for all activities. The most popular ones are Lack of need for AI assistance, AI-generated output being inaccurate, User's lack of trust and the desire to feel in control, and Lack of understanding context by AI assistant. • Areas of future improvement. Based on the prevalence of dif- ferent reasons, we formulate the main areas of future work that are needed to overcome the shortcomings that the respondents describe. This includes the improvement of base systems, better integration into developers' workflow, and more actively edu- cating and informing users about the assistants' capabilities and limitations.

Liang et al. [12] conducted an exploratory qualitative study on the usage of AI programming assistants, highlighting motivations for usage, usability challenges, and implications for creators and users of these tools. The study had 410 developers as participants. The

authors concluded that while using AI to reduce keystrokes, finish programming tasks quickly, and recall syntax, developers sometimes struggle to receive AI outputs that align with their requirements and expectations. Wang et al. [13] present a study that aims to understand practi- tioners' expectations on code completion, compare them with existing research, and highlight the need for researchers to develop techniques that meet practitioners' demands. The methodology involves semi- structured interviews with 15 professionals and an exploratory survey with 599 professionals. The authors found that practitioners expect code completion tools to work for different granularities and scenarios. They also expect a tool to be accurate, display personalized completion, be available offline, and be relatively fast. Ziegler et al. [14] discuss the use of neural code synthesis in soft- ware development and study perceived productivity by investigating whether usage measurements of developer interactions with GitHub Copilot can predict perceived productivity as reported by developers in a survey. The authors found that the acceptance rate of shown suggestions is a better predictor of perceived productivity than more specific metrics regarding the persistence of completions in the code over time. This suggests that the rate at which suggestions are accepted drives developers' perception of productivity.

Pothukuchi et al. [15] examine the potential of generative AI to automate and enhance traditional software development practices, thereby improving efficiency and reducing costs. A total of 30 profes- sionals participated in the interview study and highlighted significant improvements in development speed and code quality. The authors indicate that generative AI can significantly transform the SDLC by automating repetitive tasks and providing intelligent code sugges- tions, and propose a new model called Generative AI-Assisted Software Development Lifecycle. Mozannar et al. [16] conducted a comprehensive study to inform the understanding of how developers interact with AI tools and how to improve this experience. They studied the impact of GitHub Copilot on programmers' behavior during coding sessions. As a result, they identified 12 common programmer activities related to AI code comple- tion systems. The researchers found that developers spend more time reviewing code than writing it.

Barke et al. [17] propose a grounded theory of the bimodal nature of programmers' interactions with AI assistance that states that there are two main interaction modes between which developers fluidly switch while programming — exploration and acceleration. In acceleration mode, the programmer already knows what they want to do next, and AI helps them get there quicker; interactions in this mode are fast and do not break programmer's flow. In exploration mode, the programmer is not sure how to proceed and uses Copilot to explore their options or get a starting point for the solution; interactions in this mode are slow and deliberate, and include explicit prompting and more extensive validation.

Moreover, several companies from industry conducted large-scale studies to form an understanding of the market of AI tooling for coding. A survey of 89,184 developers conducted in 2023 by Stack Over- flow [9] found that 70expressing a favorable view. Respondents noted increased productivity with these tools, as well as their trust in their accuracy. The most recent Stack Overflow survey [18], conducted on 1700 people, revealed the continuation of the trend — users report more quality work time. A GitHub survey [19] of 500 non-manager developers found that 67believe AI coding tools will benefit their work, primarily for upskilling and productivity gains. 81oration, particularly in security reviews, planning, and pair program- ming. The JetBrains Developer Ecosystem Survey

[20] found that de- velopers are optimistic about AI advancements and actively use its capabilities despite security and ethical concerns. 59concerns, 42AI services for work. Most commonly, developers use AI to ask gen- eral software development questions and generate code, comments, or documentation.

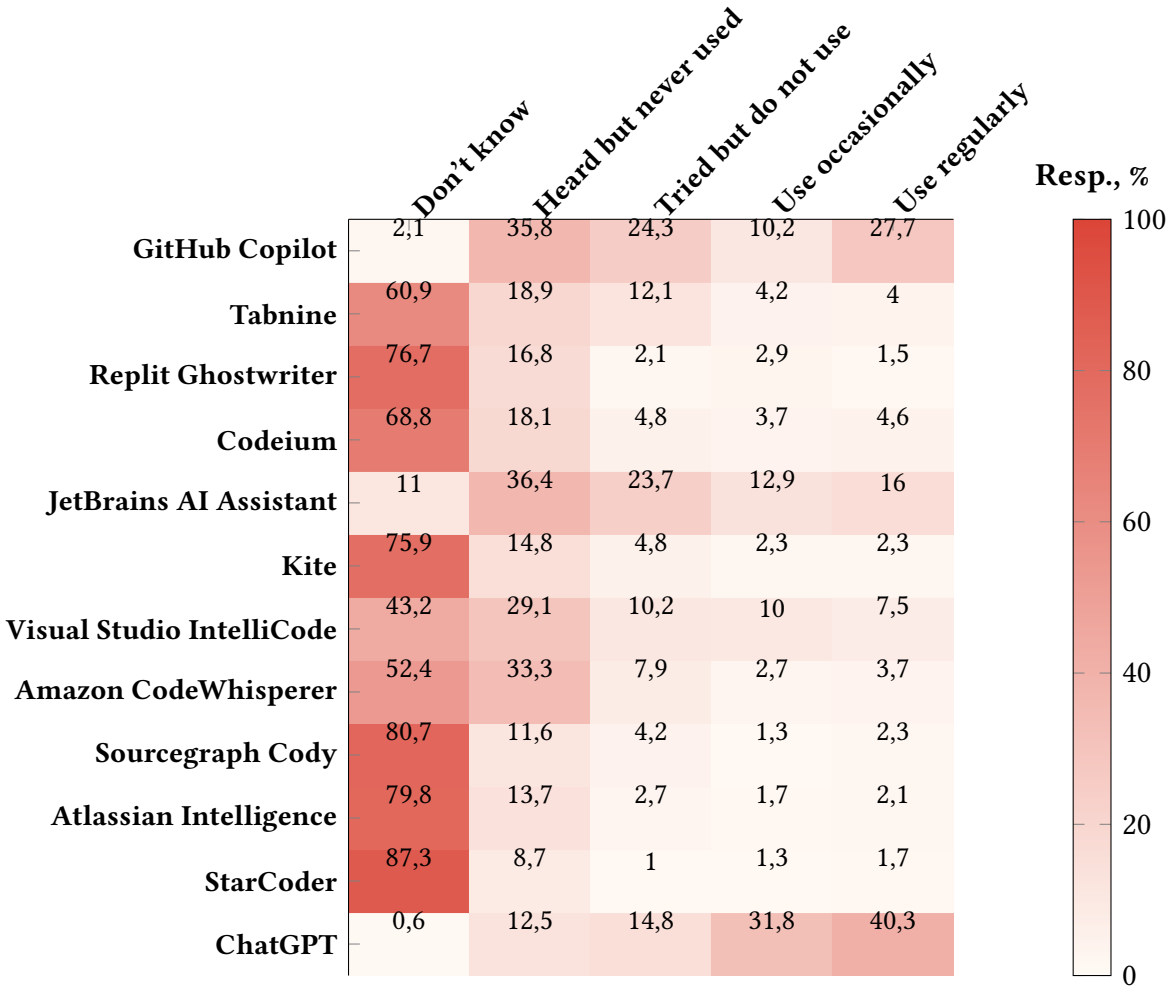

Fig. 1. The tools used by the respondents.

3.2 *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 5.

Capítulo 4

Literatura Cinzenta

Neste capítulo, estão apresentadas as fontes de literatura cinzenta utilizadas neste trabalho. Cada seção traz um resumo do conteúdo da fonte, seus resultados relevantes para este trabalho e um breve contexto da organização por trás dela.

4.1 *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.

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 4.1: *Histórico de métricas de código de 2020 a 2024 (GitClear, 2025).*

Na tabela Tabela 4.1, 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	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 4.2: *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 4.2, 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.

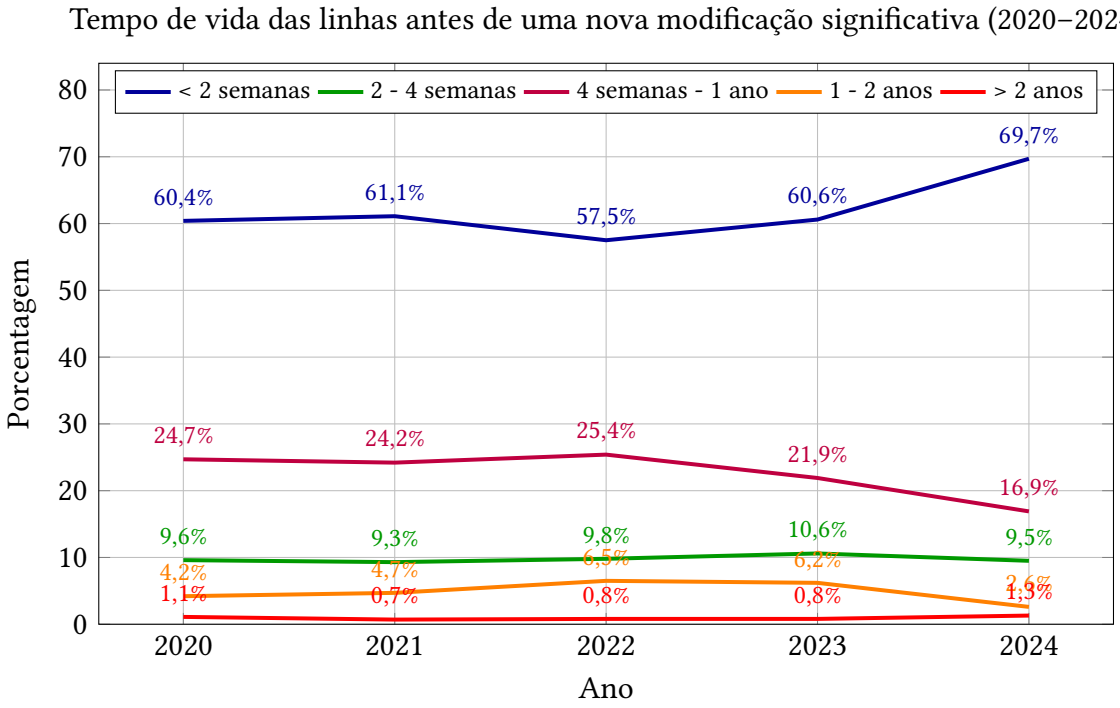

Figura 4.1: *Proporção anual das linhas modificadas classificadas pelo tempo que permaneceram no repositório antes de sofrerem uma nova modificação significativa entre 2020 e 2024 (GitClear, 2025).*

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 4.1 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 4.1, 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.

4.2 *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.

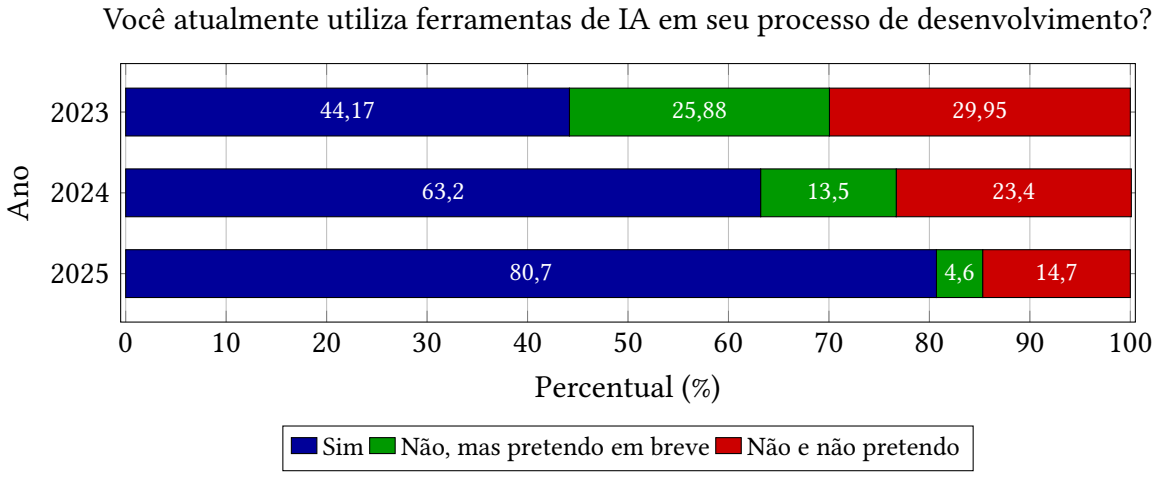

Figura 4.2: *Proporção anual dos desenvolvedores profissionais que utilizam ferramentas de IA em seu processo de desenvolvimento entre 2023 e 2025 (Stack Overflow, 2023; 2024; 2025; n_{2023} = 67237; n_{2024} = 46049; n_{2025} = 33662).*

Na Figura 4.2, 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.

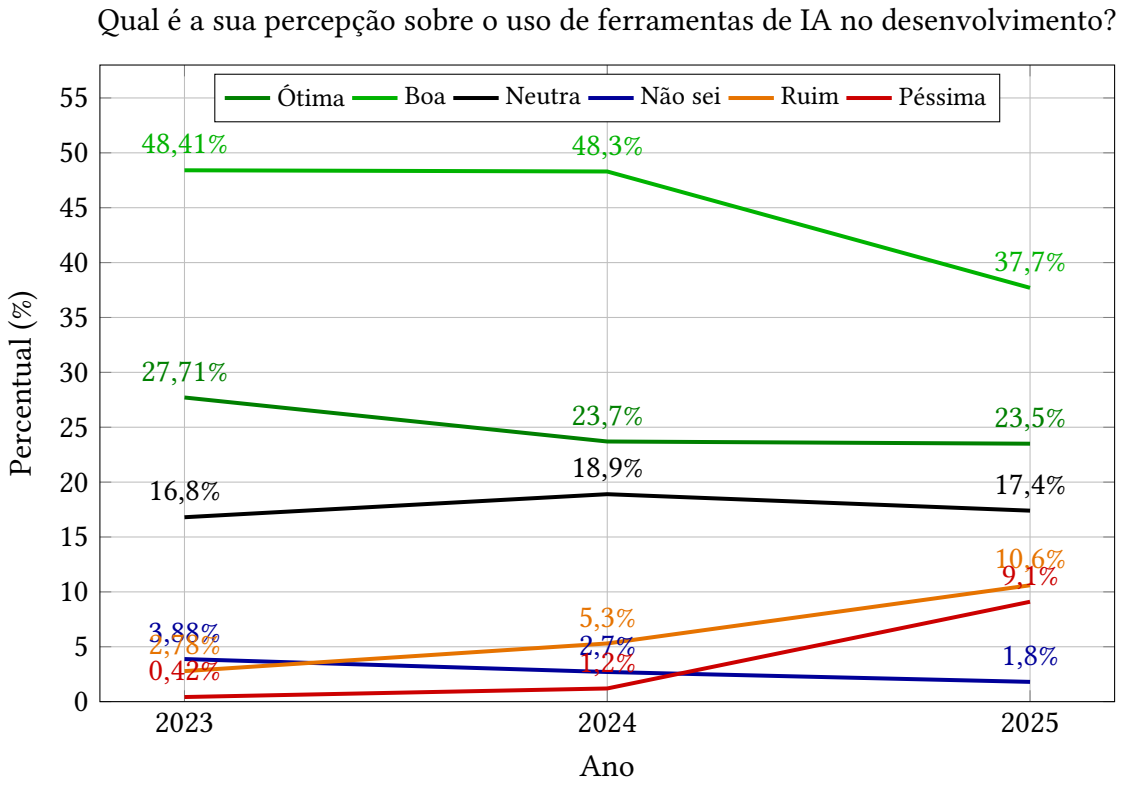

Figura 4.3: *Proporção anual da percepção dos desenvolvedores profissionais em relação às ferramentas de IA de 2023 a 2025 (Stack Overflow, 2023; 2024; 2025; n_{2023} = 46928; n_{2024} = 35142; n_{2025} = 25814).*

Na Figura 4.3, 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 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 4.4 apresenta o grau de confiança dos programadores 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

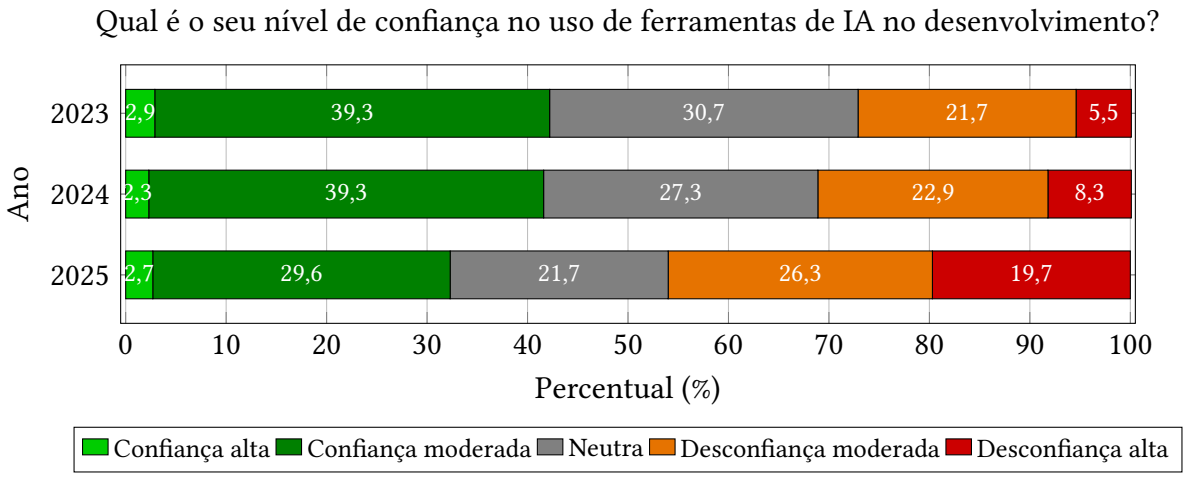

Figura 4.4: *Proporção anual do nível de confiança dos desenvolvedores em ferramentas de IA entre 2023 e 2025 (Stack Overflow, 2023; 2024; 2025; n_{2023} = 39042; n_{2024} = 28829; n_{2025} = 25701).*

nas ferramentas de IA que o contrário.

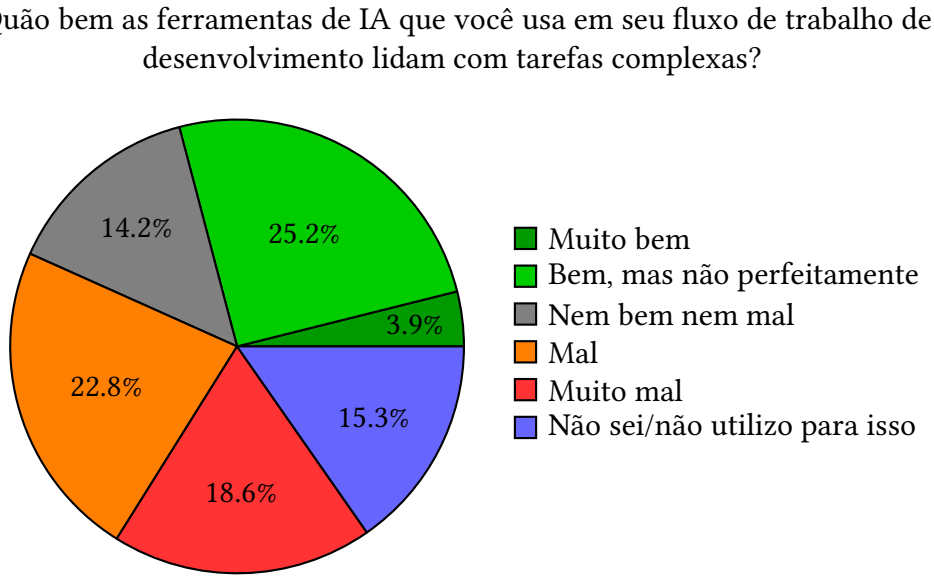

Figura 4.5: *Percepção dos desenvolvedores profissionais sobre a capacidade das ferramentas de IA em lidar com tarefas complexas em 2025 (Stack Overflow, 2025; n = 25695).*

A Figura 4.5 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.

A Figura 4.6 apresenta os principais usos que os programadores têm feito das ferramen-

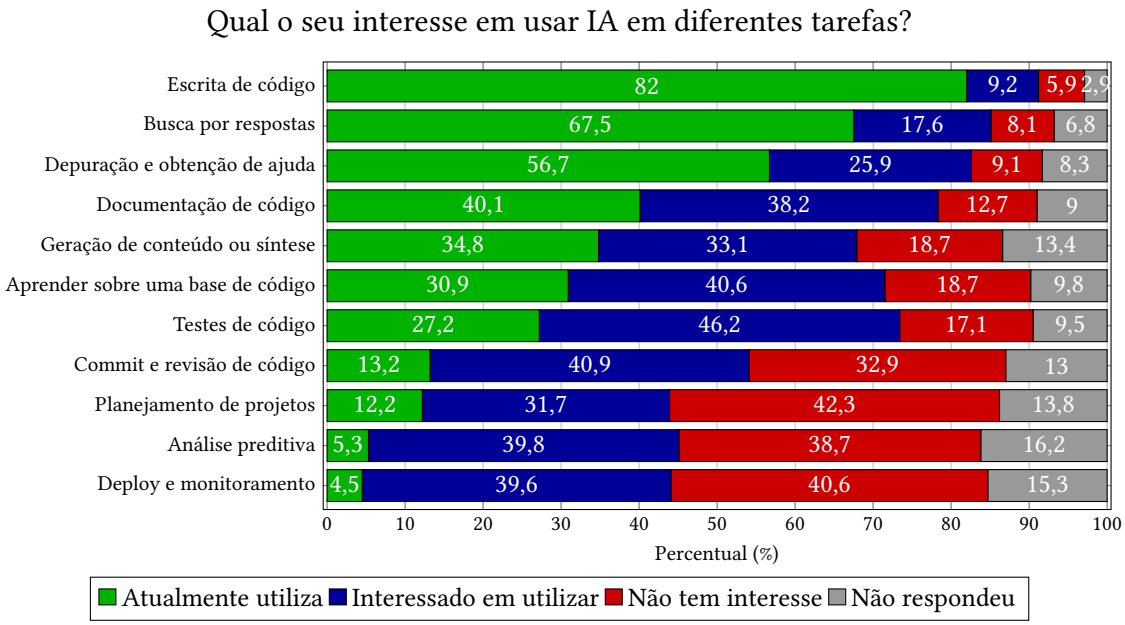

Figura 4.6: *Uso atual e interesse futuro no uso de ferramentas de IA ao longo do fluxo de desenvolvimento em 2024 (Stack Overflow, 2024; n = 35978)*

tas 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 sobrepunham e, em 2025, havia mais categorias possíveis, incluindo "usa majoritariamente IA", "parcialmente IA", entre outros, dificultando a comparação com anos anteriores.

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 4.3: *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_{2023} = 37726; n_{2024} = 35978; n_{2025} = 25349).*

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 4.3, 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.

4.3 *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 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 4.7: *Quadrante Mágico dos assistentes de código com IA (Batchu et al., 2024)*

Na Figura 4.7, 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 código aberto, 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 código aberto 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.

4.4 *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 evitará 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 1 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;

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 Figura 4.8, 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.

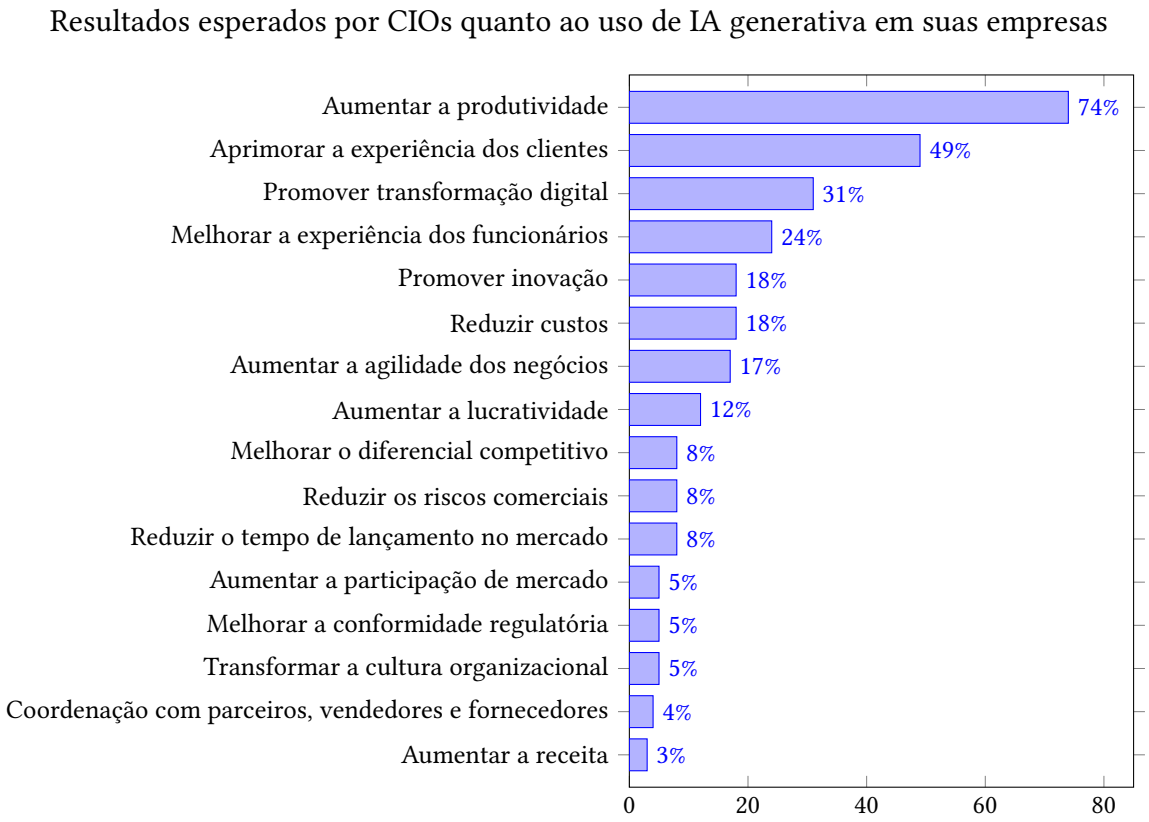

Figura 4.8: *Principais resultados esperados por CIOs nos negócios com a aplicação da IA generativa (Coshow et al., 2024)*

Capítulo 5

Análise comparativa

• produtividade e qualidade de código, • mudança no fluxo de trabalho do desenvolvedor, • confiança e percepção de precisão das ferramentas, • riscos de segurança e privacidade, • impactos educacionais e profissionais, • papel das ferramentas de geração de código.

3.4. Análise da evolução da percepção sobre IA

Um dos objetivos centrais do trabalho é traçar o amadurecimento da percepção sobre IA generativa no desenvolvimento de software. Para isso, são avaliadas mudanças no discurso, nas métricas e nas narrativas observadas nos relatórios ao longo dos últimos anos.

São analisados: • variações de adoção entre anos consecutivos, • mudanças no grau de confiança dos desenvolvedores, • expectativas declaradas para o futuro, • evolução de preocupações (por exemplo: de produtividade → segurança → governança), • transformações no perfil de usuários e suas motivações.

Diferença entre os usos que os desenvolvedores dizem que fariam com IA e o que fazem de fato relacionando isso com a satisfação (já que novas funcionalidades são a coisa preferida);

Analisar resultados que mostram que programadores se sente produtivos aceitando sugestões de ia com gartner e ia agêntica e com gartner e os erros de código não revisado.

Relacionar os diferentes indicativos de que a adoção da AI continua, gartners e overflow, dando robustez à análise e sugerindo a continuidade da tendência. Comentar também redução expressiva do tráfego do Stack Overflow que apareceu inclusive na forte diminuição de respondentes da developer survey.

Evidenciar o problema no longo prazo, que isso é só o começo da tendência.

Capítulo 6

Conclusão

3.5. Limitações da metodologia

Apesar de ampla, a metodologia apresenta limitações, entre as quais: • Dependência da literatura cinzenta, que pode conter vieses institucionais, comerciais ou metodológicos. • Falta de padronização entre reports, que utilizam diferentes métodos de coleta, tamanhos de amostra e categorias de análise. • Foco em documentos de grande impacto, o que pode excluir estudos menores ou alternativos que ofereçam visões diferentes. • Uso de um único artigo principal da literatura formal, o que, embora permita uma análise estruturada, reduz o espectro acadêmico considerado.

Essas limitações são discutidas ao final do trabalho, juntamente com sugestões de pesquisas futuras.

3.6. Possíveis continuidades

A partir da metodologia adotada, diversas linhas de continuidade são possíveis: • expansão da análise para mais artigos acadêmicos e estudos longitudinais; • avaliação empírica do uso de ferramentas de IA em ambientes reais de desenvolvimento; • investigação da adoção de LLMs em contextos educacionais; • estudo da relação entre qualidade do código gerado e maturidade das ferramentas; • incorporação de dados de novos relatórios da indústria à medida que forem publicados.

Sugerir continuidade do acompanhamento das fontes, inclusive vendo o desenrolar de tecnologias agênticas.

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 (citado nas pgs. 9, 10, 13).

[*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 Khandabattu. *Magic Quadrant for AI Code Assistants*. Rel. técn. Document 5682355. Gartner Research, 2024. URL: https://www.gartner.com/en/documents/5682355 (citado nas pgs. 9, 21, 22).

[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 (citado nas pgs. 9, 23, 25).

[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. 9, 15, 16).

[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.ieeecomputersociety.org/10.1109/ICSE.2009.5070547 (citado na pg. 16).

[*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.ieeecomputersociety.org/10.1109/MS.2024.3439138 (citado na pg. 1).

[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. 6).

[Sergeyuk *et al.* 2025] Agnia Sergeyuk, Yaroslav Golubev, Timofey Bryksin 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. 9–11).

[Stack Overflow 2023] Stack Overflow. *Stack Overflow Developer Survey*. 2023. url: https://survey.stackoverflow.co/2023/ (acesso em 19/08/2025) (citado nas pgs. 9, 17–20).

[Stack Overflow 2024] Stack Overflow. *Stack Overflow Developer Survey*. 2024. url: https://survey.stackoverflow.co/2024/ (acesso em 19/08/2025) (citado nas pgs. 9, 17–20).

[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, 9, 17–20).

[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. url: https://doi.org/10.1145/3715003 (citado na pg. 1).