Title
Early Results from a Study of GenAI Adoption in a Large Brazilian Company: The Case of Globo

ISBN
9783031556418

Authors
Pereira, Guilherme
Prikladnicki, Rafael
Jackson, Victoria
et al.

Publication Date
2024

DOI
10.1007/978-3-031-55642-5_13

Early results from a study of GenAI adoption in a large Brazilian company: the case of Globo

Guilherme Pereira[1][0009−0006−3521−6081], Rafael Prikladnicki[1][0000−0003−3351−4916], Victoria Jackson[2][0000−0002−6326−931X], André van der Hoek[2][0000−0001−7917−932X], Luciane Fortes[3], and Igor Macaubas[3]

[1] Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, RS, Brazil
[2] University of California, Irvine, Irvine, CA, USA
[3] Globo, Rio de Janeiro, Brazil

Abstract. Given the nascent and evolving nature of the latest Generative AI tools, there is little advice as to how best adopt generative AI tools within software teams or what benefits and concerns can be expected. In this chapter, we share the experiences of Globo, a large media group, that has recently begun to adopt OpenAI ChatGPT and GitHub Copilot for software development activities. We describe Globo's adoption approach and provide early insights into potential benefits and concerns in the form of eight initial lessons that are apparent from diaries kept by developers as well as semi-structured interview with them. Among the lessons learned are that the use of Generative AI tools drives the adoption of additional developer tools and that developers intentionally use ChatGPT and Copilot in a complementary manner. We hope that sharing these practical experiences will help other software teams in successfully adopting generative AI tools.

Keywords: Generative AI · Software Teams · Developer Tools · Experience Report

1 Introduction

Software teams have always adapted their ways of working in response to technological and social advances. From the adoption of agile and lean ways of working, through embracing DevOps practices, to utilizing cloud infrastructure, modern-day engineering teams have accelerated their deliveries to where development cycle times are measured in hours and days with multiple deployments to production per day [12]. Generative AI, as evidenced through tools like OpenAI ChatGPT[4] and GitHub Copilot[5], is the latest disruption that is expected to have a profound impact on software teams [6]. An often touted key benefit is an increase in developer productivity due to the ability of generative AI tools

[4] https://openai.com/chatgpt
[5] https://github.com/features/copilot

to speed up development activities, such as coding [14] and testing [9], by automatically generating code. Leveraging this automation requires developers to change the way they work; they spend less time writing code and more time reviewing and understanding code [5].

Yet adopting new ways of working and new tools is challenging for individuals, teams, and companies. It can be time-consuming and expensive to retrain teams to use new tools and to adapt their practices, especially when the tools are immature and still evolving with little practical guidance, all while ensuring that code is still shipped and deadlines are met. Companies thus need to be clear on the benefits before embarking on a potentially lengthy and costly journey of adopting generative AI tools [1].

To help others who are considering embarking on such a journey, we provide this preliminary experience report of Globo, a media group, which has recently adopted both OpenAI ChatGPT and GitHub Copilot for software development activities. We share the approach for rolling out the tools and provide early insights into the potential benefits and issues of using Generative AI. We defined the following research question: what are the main lessons learned by adopting Generative AI for software development at Globo?

The remainder of this chapter is organized as follows: section 2 presents background information, while section 3 introduces the research design. Section 4 presents the eight lessons learned and section 5 concludes the chapter.

2 Background Information

This section provides a brief overview of the capabilities, benefits, and limitations of GenAI tools used by software developers, with a focus on the two adopted by Globo: OpenAI ChatGPT and GitHub Copilot.

GenAI is artificial intelligence capable of generating text, images, or other media, using generative models [19]. These models learn patterns from input data to generate data with similar characteristics. Recent advances in neural networks have led to a number of GenAI tools. Some are general-purpose conversational agents (e.g., OpenAI ChatGPT, Google Bard[6]) while others are geared towards a specific audience such as developers (GitHub Copilot). In both cases, the models have been trained on input data scoured from many sources on the Internet, including some that are software engineering relevant (e.g., code from open-source projects maintained on GitHub).

Within software development, ChatGPT has been shown to assist developers in a variety of common software development activities, including authoring requirements [27], generating architecture [2] and design models [8], fixing defects [24], generating code [20], and seeking help [13].

Research on Copilot notes that it provides developers a perceived productivity boost, enabling them to work faster on repetitive tasks and an ability to focus on more satisfying work [14]. Code generated by Copilot is of low cyclomatic complexity [21] and is of the same complexity and as readable as human-generated

[6] https://bard.google.com/

code [3]. When given coding problems, Copilot has been shown to generate valid (e.g., compilable, interpretable) code although less than half of the solutions were correct [28]. As well as coding, Copilot can assist with authoring unit tests and identifying defective code [5].

However, it is not all positive news when it comes to using GenAI for software activities. Drawbacks of using GenAI include hallucination, as in when ChatGPT generates code that references libraries or packages that do not exist. This can provide an exploitation entry point for bad actors [16], leading to security risks [25]. Also, code generated by ChatGPT and Copilot can contain vulnerabilities [15]. The code may contain bugs that may not be immediately apparent on initial inspection [18]. In addition to these security and technical issues, there are wider societal concerns such as potential copyright issues [17], the potential for bias [7], and emerging legal issues [23].

Moreover, developers need to change their working practices to derive benefits from GenAI tools. They need to be cognizant of recommended prompt engineering techniques [26]; that is, how to best structure and organize their prompts to gain helpful answers. Also, to overcome some of the limitations of GenAI tools such as generating defective or insecure code, engineers need to review and understand the code [20].

3 Research Design

The setting of the study is the Digital Platforms structure within Globo[7], a large media group based in Latin America. This structure is part of the Digital Hub, which is Globo's Digital Technology division, and has three areas: GloboID (identity provider), Webmedia, and Publishing platform. These areas have approximately 250 people who provide solutions for digital products.

The study particularly concerned a pilot of introducing ChatGPT and Copilot to six teams with over forty employees total in various roles (e.g., developers, UX, DevOps, product owners, managers). A typical team consists of about five developers, a UX designer, and a dedicated product owner. Each team is responsible for both the development and deployment of the software it develops. Figure 1 depicts the distribution of participants based on their respective roles within the teams. Developers are predominant. The teams follow agile practices from well-known frameworks such as Scrum and XP, and they also use Kanban. Teams are self-organized and usually run monthly sprints.

Within this software division, it had been agreed with the main executives to evaluate the use of generative AI tools, specifically ChatGPT and GitHub Copilot, in software development tasks. These two tools were chosen mainly for the following reasons:

1. GitHub was already in use by the teams.
2. These two tools appear the most commonly used and talked about when Generative AI is brought into software development.

[7] https://grupoglobo.globo.com/

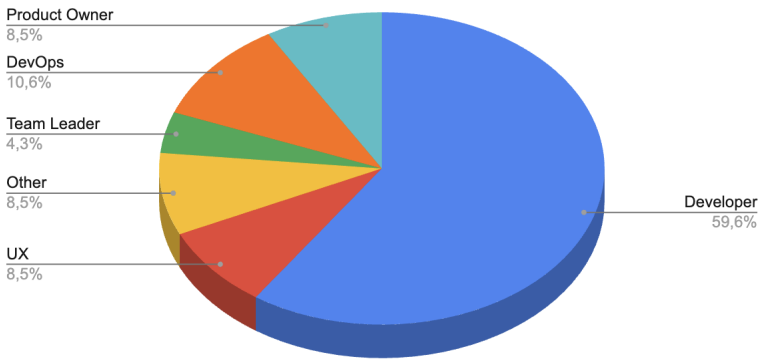

Fig. 1. The participants roles in their respective teams.

A project was initiated with the goal of rolling out the generative AI tools to the development teams while ensuring that data was captured to evaluate the benefits the AI tools provided. Specifically, the company aimed to assess the impact of using these tools in the software development process to support the decision of whether to invest in the technology more broadly across the organization.

The versions of the tools that were introduced are GitHub Copilot for Business and OpenAI ChatGPT 3.5. The company decided to use the free version of OpenAI's tool due to operational issues.

3.1 Project Approach

A multi-disciplinary team was assembled to execute the project. This team consisted of a project manager, representatives from the company's legal department, and researchers from two Universities - PUCRS and UCI (one professor and one PhD student each). Legal representatives were also included, as the company had several legal concerns about the use of generative AI. These were mostly about data leakage, code ownership, and code licenses that may govern the code being suggested by the tools.

The academic researchers' involvement in the study primarily consisted of acting as unpaid consultants to the project team. They helped on designing the study, including strategies for data collection and analysis. These also includes recommended metrics and qualitative data that would be helpful in assessing the potential benefits of the use of generative AI tools in software activities. The project consisted of three phases planned to be completed in five months from August 2023: Preparatory (1 month), Pilot (1 month) and Rollout (3 months). At the time of writing, the Rollout phase is ongoing.

The first phase was necessary to undertake any preparatory activities required by the pilot and rollout, such as license procurement and agreement on

the specific metrics to be used to understand the outcomes of the study. It included meetings with Digital Platforms management to define the study protocol, decide which teams would take part in the study, and choose which of the six teams would do the pilot before all teams participated in the rollout.

The aim of the pilot was to obtain feedback on the use of generative AI tools from a single team, so as to be able to refine and align the study protocol with actual experiences. The outcomes of this phase were presented to both the participants and the management of the company for a final round of feedback.

Globo's management selected the pilot team based on the characteristics and availability of the team. The pilot team subsequently participated in a kick-off event to familiarize its members with the project and establish the essential infrastructure. Team members were instructed to use the tools freely, but record experiences and examples of use in a spreadsheet (called a "diary" in the remainder of this chapter). Effectively, the team members were encouraged to experiment with the tools leading to experiential learning.

Participants were encouraged to use the tools freely in any software engineering activity they felt appropriate. Following the lessons learned in the pilot stage, the third phase of the project has now been kicked off with six participating teams. The same instructions were provided to these teams. This third phase is in progress.

3.2 Data Collection

Various of qualitative data was captured both during the pilot and the first few weeks of the broader rollout. Data collected during the pilot consisted of:

- At the start of the pilot, an initial survey[8] (pre survey) captured the profile of the participants and their expectations and concerns regarding the use of the tools.
- Daily, a free-style document for team members to share their day-to-day experiences with the tools (the "diary"). In this document, participants were encouraged to provide descriptions of tasks and situations in which they utilized the tools.
- Weekly, a 30-minute meeting took place between the researchers and the pilot team to collect participants' experiences of the tools that week. This included discussions on the tasks they were trying to accomplish with the AI tools and whether the tools were helpful.
- At the end of the pilot, a second survey [9] (post survey) captured participants' perceptions of the tools, including their impact on the development process.

During the rollout phase, the same data was captured as in the pilot phase. However, some adjustments were made in the diary document to collect which AI tool was used for which tasks. We also collected how participants felt about

[8] https://forms.gle/pPiF3vCAtahVrjcH9
[9] https://forms.gle/wwYiCKAg6V314KZKA

their GenAI experience on that particular day, using a three-point scale: happy, neutral, and sad.

All the members of Team 1 participated in the pilot phase. In the current stage, the initial survey was answered by 47 people distributed among the six teams, in the roles shown in Figure 1.

Due to the increase from one pilot team to six participating teams, in-person debriefs were held every 15 days on a team basis for 30 minutes. We used the following questions to guide the discussion:

- Please share the positive aspects and difficulties of interacting with ChatGPT and Copilot.
- Do you feel more productive?
- Did you identify any security threats or data leaks when you used the tools?
- Do you know and use any prompt engineering techniques?
- How has using Copilot/ChatGPT increased or reduced creativity and innovation in your coding, design, and other tasks?
- Do you think you are saving time?
- If so, what have you done with the time saved?
- How has using Copilot/ChatGPT affected your general attitude towards work, as well as your personal well-being at work?

To date, we have interviewed each team two times. Table 1 indicates the type of application the teams develop. Table 2 contains other relevant information about the teams, the technologies they use and software development processes and practices adopted.

Table 1. Project developed by teams.

Team	Application Description
T1	Develops a microfrontend platform that standardizes and facilitates the development of user flows in registration, authentication, authorization and privacy.
T2	The team works on the audio and video player for Android platforms. It plays all of globo's videos and podcasts on all the products that have apps for these platforms.
T3	The team handles the delivery of streaming videos on demand from packaging to distribution to users on our CDN.
T4	The team develops the authorization application. The systems are responsible for provisioning services to users.
T5	The Platform team responsible for digital interactivity in the company's applications, e.g., voting and comments. This is delivered to the end user and also consumed by other products.
T6	On-demand image processing system. We work with compression, cropping, resizing and applying filters.

Table 2. Way of working for each team.

Team	Number of Members	Technologies	Sw Dev Process	Roles
T1	8	Go, React, Typescript.	Kanban practices	4 Dev, 2 UX, 2 Other.
T2	10	Kotlin, Kotlin Multiplatform.	Kanban practices	7 Dev, 1 Product Owner, 1 Team Leader, 1 Other.
T3	10	Go, Ruby.	Adapted Scrum practices	5 Dev, 3 DevOps, 1 Product Owner, 1 Team Leader.
T4	8	Go, React Typescript.	Adapted Scrum practices	4 Dev, 1 DevOps, 1 Product Owner, 1 UX, 1 Other.
T5	6	Go, Lua, Python, Javascript.	Adapted Scrum practices	5 Dev, 1 UX.
T6	5	Go, Python, Javascript.	Scrum practices	3 Dev, 1 DevOps, 1 Product Owner.

3.3 Data Analysis

Using the diaries and data collected during the meetings, the qualitative data analysis process was conducted in two distinct phases by the researchers, comprising an initial screening phase and a subsequent detailed analysis phase. In both phases, one researcher examined the data to identify insights and look for themes that were subsequently discussed and reviewed with other members of the research team. To help guide the analysis, five dimensions were considered:

Developer Concerns: We sought to identify the concerns and reservations expressed by developers concerning the adoption of Generative AI tools in their daily workflow.

Positive and Negative Aspects: A central aspect of our analysis was the exploration of both positive and negative facets associated with the integration of Generative AI tools in software development, allowing for a balanced understanding of their impact.

Tasks Enhanced: We explored the specific types of tasks and software development activities where Generative AI tools proved to be particularly advantageous, discerning their areas of usefulness.

Productivity Enhancement: Lastly, we sought to evaluate the extent to which these tools contributed to overall productivity gains within the software development process.

Usage Across Roles: An integral element of the analysis was an examination of how developers and other team members across different roles were leveraging Generative AI tools, shedding light on variations in usage patterns.

This analysis resulted in the identification of the initial eight lessons described later.

3.4 Limitations and Threats to Validity

Our study is subject to some threats and limitations.First, the absence of quantitative data, such as bug counts or delivery times, restricts the ability to provide concrete, measurable insights into the impact of these tools on development processes. That is, our study shares perceptions from developers that ultimately may not be true. Complementary or quantitative studies are needed.

Second, the study's focus on a single large company may limit the generalizability of its findings. Lastly, qualitative data collection, particularly through diaries and interviews, introduces the possibility of response bias and subjectivity. While we sought to minimize this risk by comparing developers' perspectives with existing literature and interviewing a number of different teams to see if there was commonality in the perceptions across teams, such risks will always persist.

4 Eight Lessons

This section presents lessons learned from our ongoing study, until now, based on the teams described in section 3.2. By analyzing the diaries and checkpoint meetings, we have identified eight preliminary lessons about the adoption and usage of GenAI by the developers. The lessons cover a variety of software development activities. Some of the lessons apply to one of the tools only (e.g., Context Matters) whereas others apply to both tools (e.g., Improving Unit Testing). These eight lessons are shown in Figure 2 and categorized by the applicable tool (ChatGPT and/or Copilot).

4.1 Lesson 1: GenAI's Indirect Impact

At Globo, the influence of GenAI went beyond its primary functions. The development teams increasingly recognized that GenAI indirectly encouraged them to explore tools and practices that enhance code quality and overall project success. Specifically, the Globo developers started using Sonar[10] not just as a

[10] https://www.sonarsource.com/

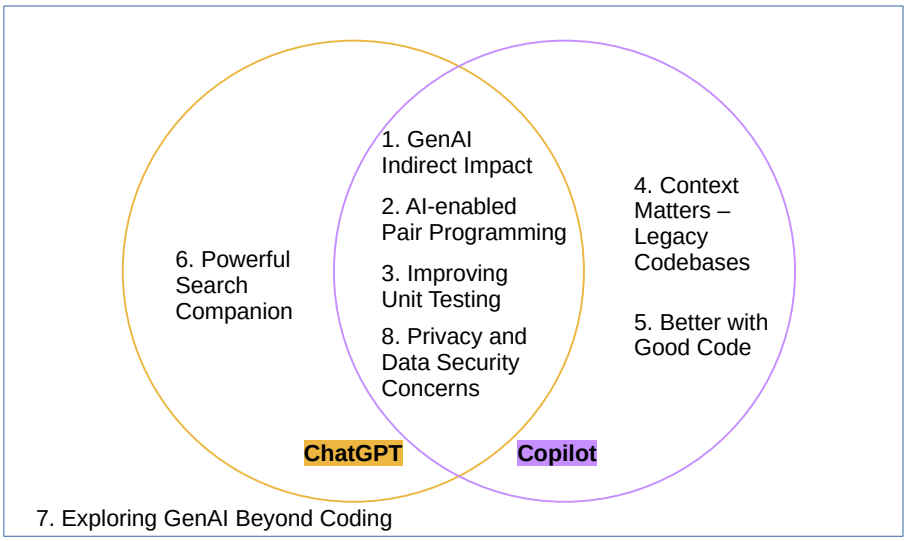

Fig. 2. Lessons learned

post-development auditing tool, but especially to collect evidence of the efficiency of GenAI (a requirement of the project). By analyzing the code generated or written with the assistance of GenAI, Sonar provided valuable insights into code quality, identified potential vulnerabilities, and highlighted areas for improvement. The developers also used ChatGPT to help using Sonar. A developer mentioned: "*I used ChatGPT to ask a question about how to use Sonar on a monorepo. The answer helped me confirm what I already thought was the way to go.* (Team 1, Dev 1)". The developers noted that the integration of Sonar into the development process enabled them to maintain a continuous focus on code quality.

4.2 Lesson 2: Learning through AI-enabled Pair Programming

Pair programming, a core practice in Extreme Programming, is a collaborative approach where two programmers work together at a single computer, with one writing code and the other reviewing it in real-time. This strategy improves code quality, reduces bugs, and promotes knowledge sharing [10]. With the advent of AI-driven programming companions like ChatGPT and Copilot, pair programming transcends human collaboration, introducing a novel dimension to coding excellence.

In our study, ChatGPT and Copilot were widely used as a pair in coding tasks like code generation and repair. The Globo developers often cited extreme pair programming [4] when describing their interaction with tools. According to them, ChatGPT and Copilot contributed in ways different from pair developers. Copilot was seen more as a smart auto-complete as it analyzes the code

in real-time, offering suggestions, auto-completing repetitive tasks, and providing context-aware code recommendations. This led to accelerated development "*Today, in pair programming with our team intern, we used Copilot to accelerate development.* (Team 1, Dev 2)". ChatGPT, on the other hand, behaved like another developer as they can direct questions to it and discuss ways to write better code. Developers engaged in a conversation with ChatGPT, and sought suggestions, clarifications, and even debugging assistance. Its ability to provide instant code snippets, explanations, and algorithmic insights mirrored the collaborative nature of pair programming. As one developer described "*We used ChatGPT to pair program, and it gave good answers to our questions.* (Team 1, Dev 3)".

One aspect worth mentioning is that the unique synergy between ChatGPT and Copilot is what truly embodies the spirit of pair programming. Developers engaged in conversations with ChatGPT for problem-solving ideas and then implemented those solutions with Copilot's real-time assistance. As one developer noted, "*Copilot helped complete the front-end code, and I asked ChatGPT a few questions about refactoring.* (Team 1, Dev 2)". Using them in tandem ensured that the code produced was not only efficient and bug-free but also inculcated a deeper understanding of programming concepts and best practices. The blend of human expertise with AI intelligence resulted in a synergy that not only enhanced code quality but also facilitated a continual learning journey for developers.

4.3 Lesson 3: Improving Unit Testing

Several developers participating in the study highlighted the usefulness of Copilot in implementing unit tests, "*Copilot helped me create unit tests in a more practical way for the code I refactored yesterday* (Team 1, Dev 1)". According to them, the tool often accelerated the development of this class of tests considerably, which seems to stem from Copilot's extensive codebase knowledge. As developers wrote code, Copilot suggested unit test cases, stubs, and mock data. It helped automate the creation of test suites, making it easier to achieve comprehensive test coverage. Globo's experience thus aligns with prior observations (e.g., [6]).

ChatGPT appeared in a somewhat different context. Globo developers used ChatGPT to understand application errors and solve problems in both production and development environments. By describing the problem in plain language, developers engaged in a conversation with ChatGPT to identify potential bug causes. For example, a critical bug in production was solved through the help of ChatGPT as "*ChatGPT helped me quickly answer questions about the (Go) language to solve an implementation.* (Team 3, Dev 1)". The developers also felt that bugs were solved quicker because ChatGPT provided richer answers than search engines "*I felt I got the answer more clearly and quickly than if I'd gone to Google directly.* (Team 5, Dev 1)".

Copilot excels in expediting the creation of unit tests, while ChatGPT's benefit lies in diagnosing the root causes of bugs, even in production environments.

For example, Copilot helped to develop unit tests with method syntax and mock-up objects, while ChatGPT was a suitable alternative to solving doubts about a specific programming language or a specific tool such as cache-control headers. Together, the combination of Copilot and ChatGPT in testing represents a potent synergy in software development.

4.4 Lesson 4: Context Matters - GenAI and Legacy Codebases

Based on the conversation with the teams, we found that Copilot's performance varied significantly depending on the context of the application it is used with. Developers working on legacy code found Copilot was not helpful while others found it more beneficial with new projects and codebases.

The integration of Copilot into projects featuring legacy code and older technology often presented developers with a unique set of challenges. Copilot's understanding of the context in such cases was less accurate due to outdated conventions, coding practices, and technology stacks. It sometimes struggled to provide relevant suggestions and did not always fully comprehend the intricacies of legacy systems. Adapting Copilot to understand and work effectively with legacy code often required manual adjustments and a deeper contextual understanding by developers. As one developer working on legacy code noted "*Copilot can't help. It doesn't seem to understand code that isn't well-structured.* (Team 1, Dev 4)". Interestingly, it appears that ChatGPT copes better with legacy code as the same developer describes "*ChatGPT gives pertinent answers to specific questions, even about legacy technologies.* (Team 1, Dev 4)".

According to the Globo developers, Copilot shined in the realm of new projects and fresh codebases. Its understanding of modern coding conventions, current technology stacks, and best practices was better. As one developer noted "*Copilot is competent to recognize context within the file.* (Team 5, Dev 1)", and another noted its support for modern-day coding practices "*Copilot looks smart for testing well-structured code. Good for isolated components.* (Team 2, Dev 1)".

Overall, this is an important point for Globo to take forward and explore, given that some of its codebase contains legacy code. A key outcome, then, is that it could be beneficial to explore how to make the GenAI tools more supportive of legacy code perhaps by building an internal language model based on its own code base. How this then balances with knowledge learned from external sources will remain to be seen.

4.5 Lesson 5: Copilot is Better With Good Code

While Copilot really helped the teams, it relied on developers crafting high-quality code to provide better suggestions. The responsibility for the quality of code suggestions was a shared endeavor that hinged on the developer's expertise. A Globo developer said that "*Copilot depends on the developer. For example, if you start writing a method with a proper name, [it] makes better suggestions. Sometimes it can't even do it.* (Team 2, Dev 1)".

Developers had to be mindful of the code they wrote, ensuring it was readable, maintainable, and followed established coding standards. Copilot, in turn, complemented the developer's work by offering suggestions that aligned with the code's context. Copilot leveraged its understanding of code patterns, best practices, and programming languages to offer suggestions. However, developers were the ones who set the foundation by writing code that was well-structured, adhered to best practices, and was clear in its intent. To illustrate this, another developer said: "*Creating a function with a descriptive name will help the Copilot answer.* (Team 3, Dev 2)".

4.6 Lesson 6: ChatGPT as a Powerful Search Companion

Some Globo developers turned to ChatGPT as a viable and favored search platform compared to Google, dedicated software engineering Q&A platforms such as Stack Overflow[11], or even documentation. They did this because ChatGPT allowed them to pose their questions in plain language and receive immediate, context-aware responses. The Globo developers felt that this type of interaction significantly reduced the time they spent searching and filtering through search results, making it an attractive choice for developers seeking swift and precise answers. Specifically, the developers noted that ChatGPT helped speed up their work as "*ChatGPT helps with technical questions faster than Google* (Team 5, Dev 2)" and "*AI delivers a more practical summary* (Team 3, Dev 2)" in comparison to traditional search engines. Moreover, the developers felt the answers were more "*assertive*" than Google searches, leading to "*accelerated work.*"

4.7 Lesson 7: Exploring GenAI's Beyond Coding

While it is clear that GenAI has found its place in coding and testing, there is a widespread lack of awareness regarding its potential in other software development activities, such as software requirements or software design. Most of the Globo reports to date concerned how developers explored several ways of generating code, obtaining debugging support, and expediting the testing process.

However, GenAI has the capability to assist beyond coding, such as in software requirements, from generating user stories and use cases to aiding in requirement analysis and documentation. One of the teams started exploring this, using it to streamline the process by seeking recommendations from ChatGPT, and ensuring that requirements are well-defined. One team member mentioned that "*ChatGPT provides good support for writing documentation. I use it a lot for documentation.* (Team 4, Dev 1)" (In this case, they mean requirements documentation, not code comments.).

GenAI can also be a good resource in design activities. It has the capability to assist in creating wireframes, generating design documents, and even automating certain design tasks. By providing design suggestions, aiding in prototype creation, and facilitating iterative design, GenAI can optimize the entire design

[11] https://stackoverflow.com/

process. At Globo, some aspects of this were explored. For example, one team member mentioned that "*ChatGPT helped us with a task that required us to build a flow diagram. It helped to elucidate complex flows in blocks of code.* (Team 4, Dev 2)."

However, our study so far showed that the use of GenAI beyond coding is underexplored at Globo. The teams have not yet explored GenAI for requirements, design, and even other activities such as software architecture. The examples mentioned above are the only instances of its use for such tasks so far. At the same time, recent literature suggests that this is common at other places too [22].

4.8 Lesson 8: Privacy and Data Security are Important Concerns

The use of GenAI brings an issue of great concern—privacy. Privacy, especially at the organizational level, is a fundamental right and a critical consideration in the age of AI. The sensitive nature of data, proprietary information, and the potential for data breaches underscore the importance of safeguarding an organization's digital assets. Based on the data collected so far, this concern with privacy limited the use and potential of GenAI in several ways. One of the primary concerns was the integration of third-party code or services, as this introduced a degree of uncertainty. This is illustrated by another quote from the study: "*Gen AI may make us use third-party code that we are not authorized to use, or we will not cite the source.* (Team 6, Dev 1)." A second concern was around data leakage, "*There is concern mainly about the leakage of credentials and/or sensitive information that may be contained in the code.* (Team 5, Dev 2)." Another developer said: "*ChatGPT needs context. It generates insecurity as more context is needed to get a satisfactory response.* (Team 4, Dev 2)." One immediate action was the development of a best practice guide by the legal team at Globo. How best to address these concerns is something to be explored further as Globo continues its adoption of GenAI.

4.9 Discussion

Although the adoption of GenAI is ongoing at Globo, these first eight lessons provide some interesting perspectives for further consideration. One noteworthy perspective is that when examining the lessons holistically, it seems that ChatGPT and Copilot are complements, rather than alternatives. Based on the analysis, they are complements for two reasons: (i) they support somewhat different use cases, and (ii) where they support the same use case, they do so in different ways by playing to each other's strengths. For example, Globo developers are using ChatGPT as a search companion (Lesson 6), because of its natural language user interface and understanding of context. For AI-enabled pair programming (Lesson 2), both ChatGPT and Copilot assisted the developer. A developer first would engage with ChatGPT to explore the problem and solution space and then complete the implementation using Copilot's intelligent auto-complete.

While Globo developers reflected on how GenAI can help them in effective ways to save time (e.g., improving testing (Lesson 3)) they also raised concerns such as the ability of GenAI to support legacy codebases (Lesson 4) and the potential for privacy risks (Lesson 8). These are crucial to address if the potential of GenAI is to be fully realized, As stated earlier, how to do so remains an open question.

The adoption of GenAI tools is also spurring Globo to adopt additional tools such as Sonar for measuring code quality and identifying potential vulnerabilities (Lesson 1). This adoption could potentially assist with some of the potential privacy and security risks noted in Lesson 8, though it is not a complete solution for that. Again, the Globo teams will need to consider the limitations and what can be done further.

Finally, it is somewhat surprising that GenAI has been little explored by Globo developers for non-coding activities such as requirements engineering or design (Lesson 7). It is unclear why this is, perhaps due to a lack of awareness of the potential of GenAI beyond merely code and test generation. Indeed, the broader research literature does not talk about this much [11], nor does the popular literature (e.g., magazines, blogs). At the same time, because of its generative capabilities, one would expect potential benefits precisely for activities such as requirements and design, where creative and broad exploration can matter. Globo's management plans to look at this potential in this regard in detail.

5 Conclusion

In the era of ever-advancing technology and the relentless evolution of software development, ChatGPT and Copilot have emerged as pivotal players, redefining the way developers write code, conduct testing, and seek answers. These AI-powered companions offered novel solutions to age-old challenges, and their presence has already left an indelible mark on the development landscape. Both tools represent a shift in how developers interact with technology, learn, and enhance their craft.

Our findings at Globo include that ChatGPT has not only expedited coding tasks but has also evolved into a rapid search companion for developers, often outperforming conventional search engines. Developers feel it offers immediate and precise information, reshaping how they access knowledge and insights.

Copilot, on the other hand, has excelled at creating synergy between human expertise and artificial intelligence, empowering Globo developers to write better code faster. Developers have recognized that Copilot's performance is closely intertwined with their own coding practices, so are cognizant of the importance of writing their own quality code to receive better suggestions.

As GenAI technology continues to evolve, it is okay to say that ChatGPT and Copilot have marked a new era that prioritizes collaboration between humans and artificial intelligence. The future of software development holds exciting potential as we continue to explore the ever-expanding horizons of AI companions

like ChatGPT and Copilot. Yet, we must proceed with care. When real production code, and especially legacy code, is involved, new practices must be found to more effectively leverage GenAI tools.

6 Acknowledgements

This work is partially supported by Globo in Brazil. Rafael Prikladnicki is partially supported by CNPq in Brazil. Guilherme Pereira is supported by the Ministry of Science, Technology, and Innovations, with resources from Law No. 8.248, dated October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex, and published in the Residência em TIC 02 - Aditivo, Official Gazette 01245.012095/2020-56. This material is based upon work supported by the US National Science Foundation under grant CCF-2210812.

References

1. Agrawal, K.P.: Towards adoption of generative ai in organizational settings. Journal of Computer Information Systems **0**(0), 1–16 (2023). https://doi.org/10.1080/08874417.2023.2240744, https://doi.org/10.1080/08874417.2023.2240744
2. Ahmad, A., Waseem, M., Liang, P., Fahmideh, M., Aktar, M.S., Mikkonen, T.: Towards Human-Bot Collaborative Software Architecting with ChatGPT. In: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering. pp. 279–285. EASE '23, Association for Computing Machinery (2023). https://doi.org/10.1145/3593434.3593468, https://dl.acm.org/doi/10.1145/3593434.3593468
3. Al Madi, N.: How Readable is Model-generated Code? Examining Readability and Visual Inspection of GitHub Copilot. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. pp. 1–5. ASE '22, Association for Computing Machinery (2023). https://doi.org/10.1145/3551349.3560438, https://dl.acm.org/doi/10.1145/3551349.3560438
4. Beck, K.: Extreme Programming Explained: Embrace Change. An Alan R. Apt Book Series, Addison-Wesley (2000), https://books.google.com.br/books?id=G8EL4H4vf7UC
5. Bird, C., Ford, D., Zimmermann, T., Forsgren, N., Kalliamvakou, E., Lowdermilk, T., Gazit, I.: Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. Queue **20**(6), Pages 10:35–Pages 10:57 (2023). https://doi.org/10.1145/3582083, https://dl.acm.org/doi/10.1145/3582083
6. Ebert, C., Louridas, P.: Generative AI for Software Practitioners. IEEE Software **40**(4), 30–38 (2023). https://doi.org/10.1109/MS.2023.3265877
7. Ernst, N.A., Bavota, G.: AI-Driven Development Is Here: Should You Worry? IEEE Software **39**(2), 106–110 (2022). https://doi.org/10.1109/MS.2021.3133805
8. Fill, H.G., Fettke, P., Köpke, J.: Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT. Enterprise Modelling and Information Systems Architectures (EMISAJ) **18**, 1–15 (2023). https://doi.org/10.18417/emisa.18.3, https://folia.unifr.ch/global/documents/324646

9. Guilherme, V., Vincenzi, A.: An initial investigation of chatgpt unit test generation capability. In: Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing. p. 15–24. SAST '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3624032.3624035, https://doi.org/10.1145/3624032.3624035
10. Hannay, J.E., Dybå, T., Arisholm, E., Sjøberg, D.I.: The effectiveness of pair programming: A meta-analysis. Information and software technology **51**(7), 1110–1122 (2009)
11. Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., Wang, H.: Large language models for software engineering: A systematic literature review (2023)
12. Humble, J., Kim, G.: Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations. IT Revolution (2018)
13. Kabir, S., Udo-Imeh, D.N., Kou, B., Zhang, T.: Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions (2023). https://doi.org/10.48550/arXiv.2308.02312, http://arxiv.org/abs/2308.02312
14. Kalliamvakou, E.: Research: Quantifying GitHub Copilot's impact on developer productivity and happiness (2022), https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/
15. Khoury, R., Avila, A.R., Brunelle, J., Camara, B.M.: How Secure is Code Generated by ChatGPT? (2023). https://doi.org/10.48550/arXiv.2304.09655, http://arxiv.org/abs/2304.09655
16. Lanyado, B.: Can you trust ChatGPT's package recommendations? (2023), https://vulcan.io/blog/ai-hallucinations-package-risk/
17. Lucchi, N.: Chatgpt: A case study on copyright challenges for generative artificial intelligence systems. European Journal of Risk Regulation p. 1–23 (2023). https://doi.org/10.1017/err.2023.59
18. Moradi Dakhel, A., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M.C., Jiang, Z.M.J.: GitHub Copilot AI pair programmer: Asset or Liability? Journal of Systems and Software **203**, 111734 (2023). https://doi.org/10.1016/j.jss.2023.111734, https://www.sciencedirect.com/science/article/pii/S0164121223001292
19. Murgia, M.: Generative AI exists because of the transformer (2023), https://ig.ft.com/generative-ai/
20. Nascimento, N., Alencar, P., Cowan, D.: Comparing Software Developers with ChatGPT: An Empirical Investigation. arXiv preprint arXiv.2305.11837 (2023). https://doi.org/10.48550/arXiv.2305.11837, http://arxiv.org/abs/2305.11837
21. Nguyen, N., Nadi, S.: An empirical evaluation of GitHub copilot's code suggestions. In: Proceedings of the 19th International Conference on Mining Software Repositories. pp. 1–5. MSR '22, Association for Computing Machinery (2022). https://doi.org/10.1145/3524842.3528470, https://dl.acm.org/doi/10.1145/3524842.3528470
22. Ozkaya, I.: Can architecture knowledge guide software development with generative ai? IEEE Software **40**(05), 4–8 (sep 2023). https://doi.org/10.1109/MS.2023.3306641
23. Ray, S.: Samsung Bans ChatGPT Among Employees After Sensitive Code Leak (2023), https://www.forbes.com/sites/siladityaray/2023/05/02/samsung-bans-chatgpt-and-other-chatbots-for-employees-after-sensitive-code-leak/

24. Sobania, D., Briesch, M., Hanna, C., Petke, J.: An Analysis of the Automatic Bug Fixing Performance of ChatGPT (2023). https://doi.org/10.48550/arXiv.2301.08653, http://arxiv.org/abs/2301.08653
25. Tal, L.: Can machines dream of secure code? From AI hallucinations to software vulnerabilities (2023), https://snyk.io/blog/ai-hallucinations/
26. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT (2023). https://doi.org/10.48550/arXiv.2302.11382, http://arxiv.org/abs/2302.11382
27. White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C.: ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design (2023). https://doi.org/10.48550/arXiv.2303.07839, http://arxiv.org/abs/2303.07839
28. Yetistiren, B., Ozsoy, I., Tuzun, E.: Assessing the quality of GitHub copilot's code generation. In: Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering. pp. 62–71. PROMISE 2022, Association for Computing Machinery (2022). https://doi.org/10.1145/3558489.3559072, https://dl.acm.org/doi/10.1145/3558489.3559072

7 Appendix - Surveys

7.1 Pre Survey

This qualitative research begins the execution of the study on the use of generative AI tools in software development at Globo. The objective of the questionnaire is to get to know the members of the teams participating in the study. Your feedback and participation are very important to understanding the characteristics, skills and perspectives of each participant. The data will only be used for research purposes.

1. Team

2. Age
a) 18 - 25 years old
b) 26 - 35 years old
c) 36 - 45 years old
d) 46 - 55 years
e) More than 56 years old

3. Gender

4. Experience in software development
a) 1 to 3 years
b) 4 to 7 years
c) 8+ years

5. Certifications in software development

6. Programming languages you've worked with
a) Python
b) Java
c) JavaScript
d) C++
e) W
f) Ruby
g) Go
h) Others

7. How long have you worked at the company? a) 1 to 3 years
b) 4 to 7 years
c) 8+ years

8. What is your role on the team?
a) Coordinator
b) Developer
c) DevOps
d) Product Owner
e) UX
f) Other

9. Do you have experience with artificial intelligence (AI) techniques?
a) No experience
b) Theoretical experiment, without application to real projects
c) Application in real projects

10. Have you used Generative AI tools (e.g. ChatGPT, GitHub Copilot)?
a) Never used
b) Yes, but not for real projects
c) Yes, even in real projects

11. Do you believe that using generative AI technologies can speed up the software development process?
a) Yes
b) No

12. Do you believe that the use of generative AI technologies can influence the adoption of best development practices?
a) Yes
b) No

13. Do you expect the use of generative AI technologies to affect collaboration and communication between development team members? How?

14. In your opinion, does the use of these tools have an impact on creativity in the software development process? How?

15. Do you have any concerns about using these technologies in professional software development?

7.2 Post survey

This qualitative research finalizes the execution of the study on the use of generative AI tools in software development at Globo. The objective of the questionnaire is to understand the perceptions of the experiment participants, after a period of using Generative AI tools. Your feedback and participation are very important to understanding the characteristics, skills and perspectives of each participant. The data will only be used for research purposes.

1. Team

2. Do you find generative AI tools useful for software development? Why?

3. What benefits (not limited to code) have you seen from using generative AI tools?

4. What negative aspects (not limited to code) have been observed in the use of generative AI tools?

5. Do you believe that generative AI tools have facilitated or accelerated the software development process? Why?

6. Have you noticed any differences in code quality when using generative AI tools? In what aspects?

7. Regarding codes written by you without the support of the AI tool, what differences do you notice?

8. Have generative AI tools helped you avoid common mistakes or identify problems in the code? In what way?

9. Do you think generative AI tools have promoted greater knowledge sharing among team members?

10. Describe difficulties or limitations you have encountered when using generative AI tools?

11. Based on your experience during the experiment, what do you think could be improved in generative AI tools?