More Productive, but at What Cost? Understanding How GenAI Shapes Developers' Work Across the SPACE Dimensions

Murilo Coelho
Atlântico Institute
State University of Ceará
Fortaleza, Ceará, Brazil
paulo_coelho@atlantico.com.br

Dione Amancio
Atlântico Institute
State University of Ceará
Fortaleza, Ceará, Brazil
dione_amancio@atlantico.com.br

Ricardo Paiva Garcia
Atlântico Institute
Fortaleza, Ceará, Brazil
ricardo_paiva@atlantico.com.br

Jose Valdir Vidal Junior
Atlântico Institute
Fortaleza, Ceará, Brazil
valdir_vidal@atlantico.com.br

Junior Almeida
Atlântico Institute
Fortaleza, Ceará, Brazil
jose_almeida@atlantico.com.br

Leandro Augusto Oliveira Leao
Atlântico Institute
Fortaleza, Ceará, Brazil
leandro_leao@atlantico.com.br

Matheus Paixao
Atlântico Institute
State University of Ceará
Fortaleza, Ceará, Brazil
matheus.paixao@uece.br

Sávio Freire
Federal Institute of Ceará
State University of Ceará
Fortaleza, Ceará, Brazil
savio.freire@ifce.edu.br

Abstract

Context. Generative Artificial Intelligence (GenAI) tools are increasingly embedded in software developers' daily work, promising productivity gains while also raising concerns about code quality, cognitive dependency, and validation overhead. However, empirical evidence is still needed to understand how these tools shape developer productivity beyond isolated measures of speed. **Goal**. This study investigates how software developers experience the use of GenAI tools in their daily work and how this use affects developer productivity across the five dimensions of the *SPACE* framework: Satisfaction and Well-being, Performance, Activity, Communication and Collaboration, and Efficiency and Flow. **Method**. We conducted a survey with 86 software developers, combining quantitative analyses of Likert-scale and frequency-based responses with qualitative content analysis of the open-ended responses. **Results**. Across the SPACE dimensions, developers perceived GenAI as a productivity accelerator. In particular, 83.7% reported increased satisfaction due to the reduction of repetitive tasks, 77.9% perceived faster delivery of valuable features, and 69.8% indicated greater confidence when tackling complex tasks. However, the findings also reveal important trade-offs: 58.1% reported subtle bugs in AI-generated code, 57.0% expressed concerns about technological dependency and skill atrophy, and 86.0% rejected deploying AI-generated code without thorough human review. **Conclusion**. GenAI improves perceived developer productivity, but its benefits are conditional on human oversight, responsible use, and organizational practices that preserve technical rigor, collaboration, and developer well-being.

Keywords

Productivity, Generative Artificial Intelligence, SPACE, Survey

1 Introduction

Generative artificial intelligence (GenAI) has attracted significant attention in software engineering (SE) research and practice due to its potential to support a wide range of software development activities, including requirements specification, design/architecture, development and testing [1, 9, 26, 30, 37, 39]. In this context, tools such as Claude [2], GitHub Copilot [18] and ChatGPT [33] have been rapidly explored in various professional software development environments [30]. Additionally, the SE community shows a growing interest in exploring the potential of GenAI tools to streamline processes and enhance productivity in software development [4, 12, 22, 24, 30, 46].

Although prior research indicates promising benefits of GenAI, such as improved productivity and faster software development [4, 8, 28, 47], Coutinho et al. [12] note that empirical evidence remains scarce, while Qian and Wexler [36] argue that it is unclear whether these systems strictly improve productivity. Furthermore, in the study performed by Vaithilingam et al. [45] participants faced difficulties in understanding, editing, and debugging code snippets generated by Copilot, suggesting that productivity gains may coexist with relevant trade-offs, such as increased validation effort and reduced understanding of AI-generated code.

Developer productivity is a broad and multifaceted concept [7, 23, 25, 32, 41]. It encompasses many aspects and therefore cannot be adequately captured by simplistic output-based measures alone [23]. In this context, productivity extends beyond code production, as developers engage in diverse activities such as providing guidance and reviewing code, designing systems and features, and managing software releases and configuration, in addition to collaborative and social tasks [23]. Then, developer productivity cannot be summarized by a single metric [49].

Understanding software developer productivity remains challenging. Forsgren et al. [17] argue that productivity is a complex and nuanced construct that cannot be reduced to a single dimension or metric. Hence, evaluating the impact of GenAI requires an analytical perspective capable of capturing both benefits and trade-offs across different aspects of developers' work. In addition,

the SPACE framework has been increasingly adopted in empirical SE research as a multidimensional lens for evaluating developer productivity [11, 16, 31]. The SPACE framework conceptualizes developer productivity as a multidimensional construct composed of complementary and potentially tensioned dimensions [17, 44, 50], comprising five dimensions: **Satisfaction and Well-being**, which captures how fulfilled and healthy developers feel in relation to their work; **Performance**, which concerns the outcomes and quality of development work; **Activity**, which refers to the count of performed actions and outputs; **Communication and Collaboration**, which encompasses how individuals and teams coordinate and share knowledge; and **Efficiency and Flow**, which represents the ability to make progress with minimal interruptions and delays. For these reasons, we adopted SPACE as our analytical lens. Unlike DevEx, which primarily focuses on developer experience [31], SPACE supports a broader, integrated analysis of key software development dimensions.

In this study, we surveyed 86 practitioners who use GenAI tools in software development activities to investigate how these tools shape developers' work and perceived productivity across SPACE's dimensions. Our study combines quantitative analyses of Likert-scale and frequency-based responses with qualitative content analysis of the open-ended responses to capture both measurable perceptions and nuanced developer experiences. Specifically, we examine the perceived benefits and concerns associated with GenAI adoption, as well as how developers relate its use to satisfaction, performance, activity patterns, collaboration, and efficiency.

Our findings indicate that GenAI acts as a potent individual accelerator across the dimensions of the SPACE framework. Specifically, 83.7% of developers reported greater satisfaction due to the reduction of repetitive tasks, 77.9% perceived faster delivery of valuable features, and 69.8% indicated increased confidence when tackling complex tasks or unfamiliar technologies. However, these benefits coexist with relevant tensions, including frequent subtle bugs in AI-generated code (58.1%), concerns regarding technological dependency and skill atrophy (57.0%), and a strong rejection of deploying AI-generated code without human review (86.0%). Overall, while GenAI leads to noticeable improvements in perceived developer productivity, these gains are conditioned upon human supervision, continuous validation, and the responsible use of the technology.

The main contributions of this study are: a multidimensional synthesis of GenAI impacts structured around the SPACE framework; an empirical study of GenAI-assisted software development using the SPACE framework in the Brazilian Software Industry; an industrial evidence from 86 developers highlighting the trade-off between faster delivery and increased technical validation overhead; and a full replication package.

2 Research Method

With the goal of investigating how developers experience the use of GenAI tools in their daily work and how this use affects developer productivity, we defined the following research questions (RQs):

- **RQ1**: *How do software developers experience the use of GenAI tools in their daily work?*
- **RQ2**: *How does the use of GenAI tools affect developer productivity across the dimensions of the SPACE framework?*

2.1 Data Collection

To collect the data, we conducted a survey regarding developers' usage of GenAI tools in their daily software development activities. A survey captures a representation of a situation in a specific context, using a questionnaire to understand a target population [48]. Thus, this method aligns with our goal of capturing the nuances of GenAI usage by software developers and how it affects their productivity. To design the questions structured under the dimensions of the SPACE framework, we built upon the research findings of Coelho et al. [11] and Ferino et al. [15]. Table 1 presents a summary of the designed questions used in this study. The complete survey can be accessed in our replication package [10].

As shown in Table 1, Q1–Q12 captured participants' profiles and experiences. Q13–Q14 addressed RQ1 by examining the perceived benefits and drawbacks of GenAI. The remaining questions addressed RQ2 across the SPACE dimensions: *Satisfaction and Well-being* (Q15–Q16), *Performance and Quality* (Q19–Q20), *Activity* (Q23–Q24), *Communication and Collaboration* (Q27–Q28), and *Efficiency* (Q31–Q32). Q15, Q19, Q27, and Q31 used a 5-point agreement scale, whereas Q23 measured the frequency of GenAI use in development tasks, such as information retrieval and code generation. It is important to note that Q17, Q18, Q21, Q22, Q25, Q26, Q29, and Q30 were not used in this research.

Next, we submitted the survey for evaluation by two experienced researchers in the field of SE. As a result of this validation, we made adjustments to the questionnaire. For example, question Q9, initially designed as a single-choice question, was modified to allow multiple choices. Following this step, we conducted a pilot study with three software developers of different seniority levels to cover a diverse range of perspectives. During the pilot, the participants did not report any ambiguities or suggest additional changes to the form. Furthermore, we observed the average time required to fully complete the survey was approximately 15 minutes.

The questionnaire was available from February 12 to March 20, 2026. We adopted a non-probability convenience sampling technique [3]. The survey was distributed through posts on professional social networks, shared within messaging app groups consisting of software developers and SE researchers, as well as through internal dissemination within partner companies and direct invitations.

At the end of the data collection period, we obtained a total of 91 responses. To ensure data validity and alignment with the research scope, we applied exclusion criteria based on Q1, Q7, and Q12. As a result, five participants were discarded: two respondents did not agree to the Informed Consent Form (Q1); two professionals were excluded because they did not perform software development activities (for instance, participant Dev_62, whose functions were limited to "requirements gathering and documentation," according to Q7; and one professional (Dev_50) was excluded for reporting no use of GenAI tools in software development (Q12). After this filtering step, we considered a final sample of 86 valid responses.

2.2 Data Analysis

The collected data were tabulated and subjected to two complementary analysis fronts: quantitative and qualitative.

2.2.1 Quantitative Analysis. Data regarding the demographic profile, work environment, and frequency of GenAI use (questions

Table 1: Survey Questions used in this Study

ID	Question Description (Q)	Type
	Informed Consent (TCLE)	
Q1	After having received clarifications about the nature, objectives, and methods of the research, do you agree to participate in this study?	Closed
	Demographic Questions	
Q2	What is your birth year?	Closed
Q3	What is your education level?	Closed
Q4	What is your undergraduate degree?	Open
Q5	How many years of experience do you have as a software developer?	Closed
Q6	Which level best describes your current professional position as a developer in terms of experience and autonomy?	Closed
Q7	Briefly describe the main activities you currently perform as a software developer.	Open
Q8	What is the total number of employees in the company you work for?	Closed
Q9	What is the domain of the software project you are currently working on?	Closed
Q10	How many people make up your direct team (squad, agile team, or working group) in the current project?	Closed
Q11	Which development process model best describes the methodology used in your current project?	Closed
Q12	What is your frequency of using GenAI tools in your software development activities?	Closed
	RQ1: GenAI Experience in Daily Work	
Q13	What are the main benefits you have perceived when using GenAI in your software development activities?	Open
Q14	What are the main disadvantages you have perceived when using GenAI in your software development activities?	Open
	RQ2: SPACE Framework Dimensions	
Q15	Rate your agreement (from 1 to 5) with statements regarding the Satisfaction and Well-being dimension.	Likert
Q16	If you wish, justify your score or include more points related to the Satisfaction and Well-being dimension.	Open
Q19	Rate your agreement (from 1 to 5) with statements regarding the Performance and Quality dimension.	Likert
Q20	If you wish, justify your score or include more points related to the Performance and Quality dimension.	Open
Q23	Considering the activities below, what is the frequency of using GenAI tools to execute them?	Frequency
Q24	If you wish, justify your choices or include more points related to the Activity dimension.	Open
Q27	Rate your agreement (from 1 to 5) with statements regarding the Communication and Collaboration dimension.	Likert
Q28	If you wish, justify your score or include more points related to the Communication and Collaboration dimension.	Open
Q31	Rate your agreement (from 1 to 5) with statements regarding the Efficiency dimension.	Likert
Q32	If you wish, justify your score or include more points related to the Efficiency dimension.	Open

Q2, Q3, Q5, Q6, Q8 to Q12) were summarized using descriptive statistics to characterize the sample. For the SPACE-related questions, Likert-scale items (Q15, Q19, Q27, and Q31) were analyzed through response frequency distributions and exploratory composite scores. The Activity dimension (Q23) was analyzed separately as a frequency-based measure, since it captures how often developers use GenAI across typical development activities. These summary scores were used to support exploratory stratified comparisons.

2.2.2 Qualitative Analysis. To process the responses from the open-ended questions, we employed content analysis [27], following an open coding process inspired by Strauss et al. [42]. This process was conducted in a purely inductive manner, ensuring that no predefined codes were imposed on the data. Initially, the researchers formed pairs and independently read the responses to the open-ended questions (Q13, Q14, Q16, Q20, Q24, Q28, and Q32) to identify codes relevant to answering the Research Questions. Calibration meetings were held after analyzing the first 10 responses to maintain alignment and consistency in the coding process. These codes were stored in a shared spreadsheet. Inter-rater agreement was assessed using Cohen's Kappa before the consensus meetings (k = 0.811). Subsequently, the pairs held iterative meetings to compare, consolidate, and refine the extracted codes, resolving any divergences in interpretation through consensus. A third, more experienced reviewer conducted a final analysis of the consolidated codes, suggesting improvements in the codes. Additional consensus meetings were held to consolidate the final result. After the coding consolidation, one researcher grouped the generated codes into higher-level thematic classifications based on the semantic similarity of the developers' reports. These classifications were reviewed by two other researchers, who assessed the consistency and representativeness of the categories. Any disagreements or refinements were discussed until consensus was reached.

For each research question, we describe the codes that emerged from our qualitative analysis, supported by representative quotes from the participants. Participant anonymity was strictly maintained through the use of pseudonyms (e.g., Dev_01, Dev_02, and so forth). Furthermore, confidential information was sanitized by redacting sensitive data, such as company names occasionally mentioned by the developers in their responses. This approach ensures adherence to ethical research practices while providing the necessary context for the accurate interpretation of the presented quotes.

3 Results

3.1 Demographics

The survey included 86 software developers from the Brazilian software industry. Most participants were born between 1981–1996 (52.3%) or 1997–2012 (41.9%), while 4.7% were born between 1965–1980 and 1.2% between 1946–1964.

Regarding education, 30.2% had completed an undergraduate degree and 23.3% were enrolled in one. Professional postgraduate degrees had been completed by 17.4% and were being pursued by 4.7%. Additionally, 8.1% held a master's degree and 9.3% were enrolled in a master's program, while 2.3% held a Ph.D. and 3.5% were pursuing one. Only 1.2% reported high school as their highest education level. Most participants (86%) had undergraduate backgrounds

in IT-related fields, while 14% came from other areas, including electrical, mechanical, electronic, and agronomic engineering.

Half of the participants had more than six years of development experience, 33.7% had two to six years, and 16.3% had less than two years. Regarding seniority, 41.9% identified as senior, 34.9% as mid-level, 17.4% as junior, and 5.8% as interns or trainees.

AI/Data Science and Retail/E-commerce were the most represented domains (24.7% each), followed by Finance/Banking (16.5%), Government/Public Sector (14.1%), Education (10.6%), and Telecom, Industrial Automation/IoT, and Healthcare (8.2% each). Participants also represented several other domains.

Most participants worked in large companies (more than 500 employees; 66.3%), followed by medium-sized (100–499 employees; 12.2%), small (20–99 employees; 10.5%), and micro-enterprises (up to 19 employees; 4.7%). Another 4.7% were freelancers, and 1.2% were unemployed. Team sizes were predominantly 6–9 members (45.3%) or 2–5 members (27.9%), followed by 10–20 members (19.8%). Only 5.8% worked alone, and 1.2% worked in teams larger than 20 members.

Finally, 58.1% used GenAI tools more than once a day, 27.9% three to five times per week, 11.6% once or twice per week, and 2.3% less than once per week. Additional demographics concerning development processes and project characteristics are available in the replication package [10].

3.2 RQ1: Software Developers' Experiences with GenAI Tools in Daily Work

To answer this RQ, we categorized participants' reports into perceived benefits and concerns associated with daily GenAI use.

3.2.1 Benefits of Using GenAI Tools. In total, we identified 207 mentions of benefits related to the use of GenAI tools. After the coding and normalization process, we consolidated the results into 59 unique benefits. Next, we grouped the benefits by thematic affinity into 7 distinct categories, as can be observed in Table 2. It shows that **Productivity** and **Technical Understanding & Learning** are the most prominent benefit, while specific operational tasks, such as **debugging** and **documentation**, were reported less frequently. Overall, the main trend indicates that developers perceive GenAI primarily as a delivery accelerator and a cognitive assistant for routine tasks, rather than a driver for high-level technical decision-making.

Table 2: Categories of perceived benefits of using GenAI tools

	Benefit Categories	#SC (%)	#BM (%)	%BM/Dev
1st	Productivity	14 (23.7%)	60 (29.0%)	69.8%
2nd	Technical Understanding & Learning	15 (25.4%)	36 (17.4%)	41.9%
3rd	Code Quality & Maintenance	9 (15.3%)	33 (15.9%)	38.4%
4th	Support for Development & Implementation	9 (15.3%)	30 (14.5%)	34.9%
5th	Debugging & Problem Solving	5 (8.5%)	26 (12.6%)	30.2%
6th	Technical Communication & Documentation	3 (5.1%)	15 (7.2%)	17.4%
7th	Support for Technical Decision Making	4 (6.8%)	7 (3.4%)	8.1%
	Total	**59 (100.0%)**	**207 (100.0%)**	**240.7%**

Caption:
#SC (%): Count and percentage of unique Standard Codes per category (N=59).
#BM (%): Count and percentage of Benefit Mentions over all mentions (N=207).
%BM/Dev: Percentage of #BM in relation to the number of participants (N=86).

Productivity, in the context of this work, encompasses benefits that focus on increasing agility and the speed of final delivery of activities, as exemplified by Dev_59: "It helps a lot to speed up the development process and the structuring of deliveries, carrying out a good part of the initial development necessary to add the business rule or feature itself." The **Technical Understanding & Learning** category consolidates benefits that seek to use GenAI to comprehend, clarify, and support learning, exemplified by Dev_54: "GenAI has helped me mainly in refining my projects [...]. It also contributes significantly to my learning, facilitating the understanding of concepts, good practices, and helping to clarify doubts during development." **Code Quality & Maintenance** encompasses benefits related to code quality and maintainability, for example: "In tedious or repetitive activities, to suggest corrections and improvements in code, code review." (Dev_78)

The benefits of **Support for Development & Implementation** incorporate coding support practices, as pointed out by Dev_30: "It helps me when I have doubts, find better solutions, resolve bugs, and develop software more effectively."

Debugging & Problem Solving represents benefits that support the resolution of errors and bugs, as reported by Dev_14: "In my perception, it has helped a lot in problem-solving and debugging [...]." For **Technical Communication & Documentation**, benefits related to communication support and artifact creation are grouped: "[...] Furthermore, it creates well-structured documentation that would also take time to be made." (Dev_82) Finally, **Support for Technical Decision Making** involves benefits that assist in decision-making, as explained by Dev_26: "[...] I also use it to help me with decisions within my own code, especially regarding software architecture."

Finding #1. The primary perceived benefit of using GenAI is the gain in **Productivity** (69.8%), highlighting its perceived value as a delivery accelerator. Beyond speed, these tools appear to serve as cognitive and structural support, with notable impacts on **Technical Understanding & Learning** (41.9%) and **Code Quality & Maintenance** (38.4%).

3.2.2 Concerns of Using GenAI Tools. In total, we identified 181 mentions of concerns related to the use of GenAI tools. We used the same code consolidation and categorization procedure as reported in Section 3.2.1. The results are reported in Table 3.

Table 3: Categories of concerns when using GenAI tools

	Concern Categories	#SC (%)	#CM (%)	%CM/Dev
1st	Degradation of Technical & Response Quality	17 (21.8%)	43 (23.8%)	50.0%
2nd	Limitations & Inadequate Behaviors of GenAI	13 (16.7%)	38 (21.0%)	44.2%
3rd	Negative Cognitive Impacts	15 (19.2%)	33 (18.2%)	38.4%
4th	Dependency, Distrust, & Dysfunctional Use	8 (10.3%)	23 (12.7%)	26.7%
5th	Overhead of Review, Validation, & Rework	9 (11.5%)	22 (12.2%)	25.6%
6th	Organizational, Procedural, & Governance Impacts	11 (14.1%)	17 (9.4%)	19.8%
7th	Behavioral & Emotional Impacts	5 (6.4%)	5 (2.8%)	5.8%
	Total	**78 (100.0%)**	**181 (100.0%)**	**210.5%**

Caption:
#SC (%): Count and percentage of unique Standard Codes per category (N=78).
#CM (%): Count and percentage of Concern Mentions over all mentions (N=181).
%CM/Dev: Percentage of #CM in relation to the number of participants (N=86).

Table 3 reveals that developers' primary concerns center on the **Degradation of Technical Quality** and the inherent **Limitations of GenAI**. In addition, human-centric apprehensions, such as **Negative Cognitive Impacts**, **Dependency**, and **Validation Overhead**, were widely reported. This trend suggests that while GenAI

accelerates output, it introduces new sociotechnical challenges, shifting the developer's effort from writing code to rigorously validating it, while raising alarms about long-term skill atrophy.

The **Degradation of Technical & Response Quality** category concerns the quality of code and activities generated by GenAI tools, as exemplified by Dev_42: "AI-generated code still contains errors and often the models make decisions that can harm the project, so constant review is still necessary." Meanwhile, **Limitations & Inadequate Behaviors** of GenAI groups concerning behaviors of the tools, such as hallucinations or improper code modifications, exemplified by Dev_13: "It hallucinates a lot or the code is of low quality. Most of the time it is better for me to develop the code myself." The **Negative Cognitive Impacts** category encompasses concerns about the loss of critical thinking, as noted by Dev_44: "When we ask for solutions to problems we asked the GenAI itself to find, we become dependent on it and stop thinking for ourselves."

Dependency, Distrust, & Dysfunctional Use reflects the fear of excessive reliance on GenAI, as explicitly stated by Dev_56: "I feel very dependent on AI now that I use it a lot." For **Overhead of Review, Validation, & Rework**, concerns related to the additional effort required for code review and validation are gathered, for instance: "need for validation/verification of the answers." (Dev_07) **Organizational, Procedural, & Governance Impacts** involves systemic issues related to the project and the organization itself, such as security risks, pointed out by Dev_82: "[...] Another disadvantage is the security risks, especially regarding the use of data by AIs, when there is no clarity on compliance." Finally, **Behavioral & Emotional Impacts** encompasses concerns regarding the human aspect and developer well-being, as reported by Dev_65: "exponential increase in anxiety for results [...] (anxiety both personally and from management)."

Finding #2. The primary concern regarding GenAI usage is the **Degradation of Technical & Response Quality** (50.0%), coupled with the inherent limitations of the tools (44.2%). This scenario not only demands constant review efforts but also brings attention to **Negative Cognitive Impacts** (38.4%), with developers expressing concerns over a potential loss of critical thinking.

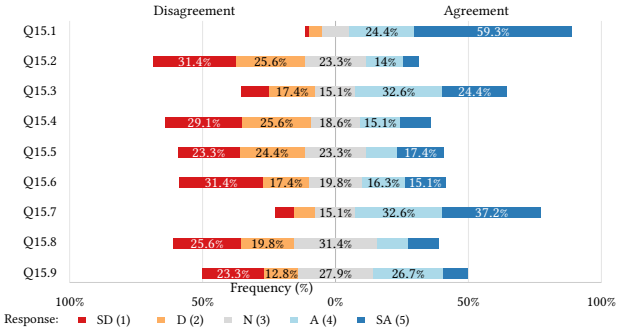
3.3 RQ2: GenAI Tool Effects on Developer Productivity Across the SPACE Dimensions

For this RQ, we structured our findings based on the dimensions of the SPACE framework: *Satisfaction & Well-being*, *Performance & Quality*, *Activity*, *Communication & Collaboration*, and *Efficiency*.

3.3.1 Satisfaction & Well-Being. To evaluate this dimension, we presented nine statements to the developers using a five-point Likert scale (where 1 indicates strongly disagree and 5 indicates strongly agree). The statements, as well as the data consolidated by response frequency ($N = 86$), are detailed in Table 4.

Regarding **satisfaction from reduced repetitive tasks** (Q15.1), 83.7% agreed or strongly agreed, 10.5% were neutral, and 5.8% disagreed. For **increased anxiety during activities** (Q15.2), 57.0% disagreed, 23.3% were neutral, and 19.8% agreed. Concerning **fear of dependency and loss of fundamental skills** (Q15.3), 57.0% agreed or strongly agreed, 27.9% disagreed, and 15.1% were neutral.

Table 4: Distribution of responses for the *Satisfaction and Well-being* dimension.

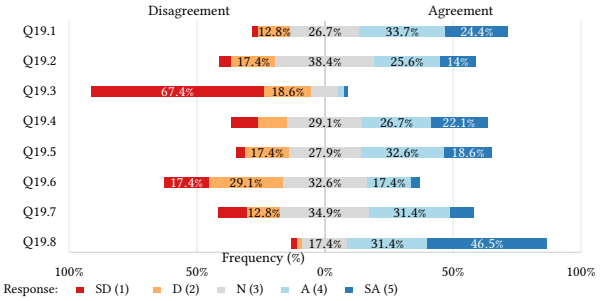


Response scale: SD = Strongly disagree; D = Disagree; N = Neutral; A = Agree; SA = Strongly agree. *Items.* **Q15.1**: The use of GenAI makes my daily work more satisfying by reducing repetitive tasks. **Q15.2**: I feel that the use of GenAI has increased my anxiety when performing my activities. **Q15.3**: I feel that the use of GenAI has increased my fear of becoming dependent on the tool (e.g., losing fundamental skills). **Q15.4**: I feel pressure from my direct management to use GenAI to increase delivery speed. **Q15.5**: I feel pressure from the company I work for to use GenAI to increase delivery speed. **Q15.6**: I feel pressure from the client to use GenAI to increase delivery speed. **Q15.7**: The use of GenAI tools has increased my confidence to take on tasks or technologies that I previously considered too complex. **Q15.8**: The availability of GenAI tools in my work environment is an important factor in my staying with the company. **Q15.9**: The availability of GenAI tools in my work environment is an important factor in my satisfaction with my role.

Developers generally did not feel strong external pressure to use GenAI. Most reported little pressure from direct managers (Q15.4, 54.7% disagreed), companies (Q15.5, 47.7% disagreed), or clients (Q15.6, 48.8% disagreed), although a noticeable minority perceived some pressure (26.7%, 29.1%, and 31.4%, respectively), and a portion remained neutral (18.6%, 23.3%, and 19.8%). In contrast, GenAI positively influences developers' confidence, with many reporting increased confidence in handling complex tasks or technologies (Q15.7, 69.8% agreed), while smaller groups were neutral or disagreed (15.1% each). However, its impact on retention and job satisfaction is less clear. Most developers do not see GenAI as a factor for staying in a company (Q15.8, 45.4% disagreed, 31.4% neutral, 23.2% agreed), and perceptions of role satisfaction are divided (Q15.9, 36.0% agreed, 36.0% disagreed, 27.9% neutral).

We requested a justification or additional comments from the developers regarding the Likert scale statements (Q16). In total, 22 points were raised by the participants, which were grouped into 21 unique additional comments. Next, we grouped the comments into seven distinct categories. The most prominent categories were **Productivity & Efficiency Perception**, **States of Well-being & Psychological Safety**, and **Cognitive Impacts on Well-being**. The first highlights improvements in productivity and value delivery, as illustrated by Dev_54: "[...] Having support to review ideas and validate technical decisions reduces insecurities and makes the development process smoother and more productive." The second reflects positive aspects of the work environment and psychological safety, for example: "My satisfaction is directly linked to the project itself. Organization, reasonable schedules, challenge and, obviously, flexibility regarding the work regime (remote)." (Dev_39). The third captures concerns about cognitive impacts, particularly dependency on GenAI, as noted by Dev_11: "I think AI is a positive point, but I don't think it's cool to depend 100% on it for everything [...] we get used to running to the AI and asking it to solve it. I don't think this is entirely positive."

Table 5: Distribution of responses for the *Performance & Quality* dimension.



Response scale: SD = Strongly disagree; D = Disagree; N = Neutral; A = Agree; SA = Strongly agree. *Items*. **Q19.1**: GenAI-generated code frequently contains subtle bugs that require my manual correction. **Q19.2**: GenAI-generated code frequently contains security flaws that may compromise the correct functioning of the system. **Q19.3**: I feel confident deploying GenAI-generated code to production environments without the need for thorough human review. **Q19.4**: The use of GenAI has improved the structural quality of my code (e.g., cleaner, better documented, or standards-compliant code). **Q19.5**: GenAI-generated code frequently contains unnecessary complexity, dead code, or duplicated code that requires my manual cleanup. **Q19.6**: I feel that constant use of GenAI has reduced my critical ability to detect logic or security errors without the tool's assistance. **Q19.7**: The use of GenAI has helped reduce the number of defects (bugs) that reach the testing or review phase. **Q19.8**: The use of GenAI allows me to deliver valuable features to the end customer faster (time-to-market).

The remaining categories provide complementary perspectives on developers' experiences. **Organizational Context and Working Conditions** reflects the influence of organizational practices and working conditions on developers' well-being. **Delivery Pressure and Dynamics** captures concerns related to delivery pressure and productivity-oriented work practices. **Support for Development and AI Use** highlights the role of GenAI in supporting software development activities and resolving technical questions. Finally, **Cognitive Effort and Validation Overhead** emphasizes the effort required to validate AI-assisted technical decisions.

Finding #3. GenAI is reported as a strong driver for operational satisfaction (83.7%) and technical confidence (69.8%) by reducing repetitive tasks and supporting complex activities. However, this positive effect is moderated by dependency concerns (57.0%), delivery pressure, organizational context, and the need for continuous technical validation of outputs.

3.3.2 Performance & Quality. We also adopted the same procedure as in the previous dimension. The results are detailed in Table 5.

Developers report mixed effects of GenAI on code quality and reliability. Many perceive that GenAI can introduce subtle bugs (Q19.1, 58.1% agreed), and a considerable portion also notes potential security issues in generated code (Q19.2, 39.6% agreed, with many neutral at 38.4%). At the same time, there is a strong consensus against trusting AI-generated code without human oversight (Q19.3, 86.0% disagreed). Despite these concerns, nearly half of the participants perceive improvements in structural code quality after adopting GenAI (Q19.4, 48.8% agreed), although a relevant share remains neutral or disagrees.

Additional perceptions reinforce this balance between benefits and drawbacks. Many developers observe unnecessary technical elements in generated code, such as accidental complexity or dead code (Q19.5, 51.2% agreed). However, most do not believe that frequent GenAI use significantly reduces their ability to detect errors (Q19.6, 46.5% disagreed). Regarding defect rates, opinions are divided, with some reporting reductions (Q19.7, 40.7% agreed) and many remaining neutral. In contrast, there is strong agreement that GenAI helps deliver value faster to customers (Q19.8, 77.9% agreed), highlighting its perceived impact on development speed despite quality-related concerns.

The justifications and additional points resulted in 20 unique comments that were grouped into six distinct categories. **Code & Project Quality** highlights perceived improvements in code quality with GenAI, particularly in patterns and readability, as noted by Dev_38: "[...] quality is often associated with patterns and readability, attributes that code GenAIs are very good at." Closely related, **Productivity & Performance Gains** emphasizes improvements in development speed and overall outcomes, with Dev_54 stating: "GenAI contributed to the decrease in development time and the increase in project quality. It assists in structure organization, code review, and identifying improvements, which reduces rework and enhances the clarity of the implemented solutions." At the same time, **Changes in the Way of Working** reflects adjustments in development practices, especially the need for careful validation of AI-generated code, as illustrated by Dev_04: "regarding performance, AI definitely helped to improve it. regarding quality, it is necessary for the developer to analyze EVERYTHING that is suggested. because the paths to be followed depend on the person who is there doing it."

The remaining categories provide a complementary perspective. **AI Limitations & Negative Project Impacts** highlights constraints and risks associated with GenAI use, while **Support for Learning & Development** points to its role in facilitating the acquisition of new knowledge. Finally, **General Perception of AI Use** reflects an overall positive view of GenAI tools in the development process.

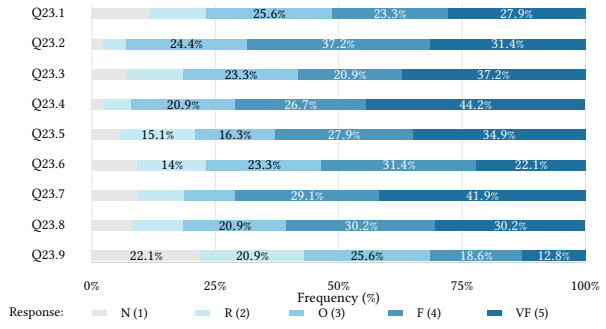
Finding #4. GenAI is widely seen as accelerating delivery (77.9%) and improving code quality (48.8%), especially readability and design patterns. However, these benefits require careful review, as AI-generated outputs may introduce subtle bugs (58.1%) and accidental complexity (51.2%), making human validation essential before deployment (86.0%).

3.3.3 Activity. To evaluate this dimension, we asked developers to indicate the frequency of GenAI tool usage across nine typical activities, using a five-point scale (from "Never" to "Very Frequently"). Table 6 details these activities, arranged according to the response frequency of the 86 participating professionals ($N = 86$).

Table 6 shows that developers commonly use GenAI across several activities, especially those directly related to coding. Frequent use is particularly high for code explanation (Q23.4, 70.9%), writing technical documentation (Q23.7, 71.0%), and code generation (Q23.2, 68.6%), indicating that GenAI is widely integrated into tasks that involve producing or understanding code. Similarly, many developers rely on GenAI for testing (Q23.5, 62.8%), code review (Q23.8, 60.4%), debugging (Q23.3, 58.1%), and refactoring (Q23.6, 53.5%), suggesting consistent support across core development activities.

Usage is moderate for information retrieval (Q23.1), where about half of the participants report frequent use (51.2%), while a considerable portion uses it only occasionally or rarely (37.2%) or not at all (11.6%). In contrast, management-related activities (Q23.9) show the lowest engagement, with fewer developers reporting frequent

Table 6: Distribution of responses for the *Activity* dimension.



Response scale: N = Never; R = Rarely; O = Occasionally; F = Frequently; VF = Very frequently. *Items.* **Q23.1**: Information retrieval (searching docs/libraries). **Q23.2**: Code generation (boilerplate/functions). **Q23.3**: Debugging (fixing errors). **Q23.4**: Code explanation (understanding legacy code). **Q23.5**: Testing (unit tests, test data). **Q23.6**: Refactoring. **Q23.7**: Writing technical documentation. **Q23.8**: Code review. **Q23.9**: Management (estimation, prioritization, general documentation).

use (31.4%) and most relying on GenAI only occasionally or rarely (46.5%) or not using it (22.1%). Overall, these results indicate that GenAI is primarily used to support technical and coding-related tasks, while its adoption in managerial activities remains limited.

We requested justifications or additional activities for the Activity dimension (Q24) from the developers. In total, four points were individually raised (1.2% each): **Log analysis**, **Documentation improvement**, **Organization and expression of ideas**, and **Use across all software development stages**. As an example of the latter category, Dev_60 stated: "My work [is] 100% with the use of AI, in all stages we use activity specifications."

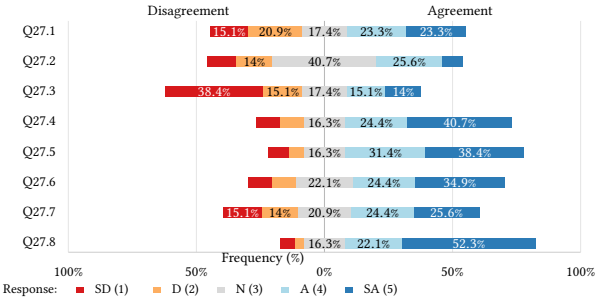
Finding #5. GenAI usage is widespread throughout the development lifecycle, but its maximum intensity occurs in knowledge-support tasks (documentation and explanation) and code generation (boilerplate). Management activities remain the least adopted context, with low usage (31.4%), suggesting that the tool serves predominantly as an operational technical assistant.

3.3.4 Communication & Collaboration. Table 7 shows that developers report mixed effects of GenAI on collaboration and teamwork. Many feel that GenAI can reduce the need to seek help from experienced colleagues (Q27.1, 46.6% agreed), although a substantial portion still disagrees (36.0%) or remains neutral (17.4%). Perceptions are less clear regarding the quality of AI-assisted pull requests, with developers uncertain about logical errors (Q27.2, 40.7% neutral, 33.7% agreed). Sharing prompts or strategies with teammates is not yet common practice (Q27.3, 53.5% disagreed), suggesting limited knowledge exchange around GenAI usage.

On the other hand, developers generally hold positive views about the broader collaborative benefits of GenAI. Most recommend these tools to others (Q27.4, 65.1% agreed) and find them helpful for understanding legacy code (Q27.5, 69.8% agreed) and improving team documentation (Q27.6, 59.3% agreed). About half report using GenAI to support code reviews (Q27.7, 50.0% agreed). The strongest agreement appears in onboarding and learning, where many developers feel that GenAI accelerates their ability to become independent in new projects (Q27.8, 74.4% agreed).

The justifications and additional points resulted in 11 unique comments that were grouped into four distinct categories. The

Table 7: Distribution of responses for the *Communication & Collaboration* dimension.



Response scale: SD = Strongly disagree; D = Disagree; N = Neutral; A = Agree; SA = Strongly agree. *Items.* **Q27.1**: The use of GenAI has replaced my need to seek help from more experienced colleagues or mentors when I have questions. **Q27.2**: I notice that the team's pull requests (PRs), when generated by GenAI, look correct at first glance but often contain subtle logical errors. **Q27.3**: I often share effective prompts or GenAI usage strategies with members of my team. **Q27.4**: I recommend the use of GenAI tools to other developers. **Q27.5**: The use of GenAI has helped me understand legacy codebases, reducing the need for detailed explanations from colleagues. **Q27.6**: The quality of the team's technical documentation has improved with the use of AI assistants (e.g., clearer pull request descriptions or useful comments). **Q27.7**: I regularly use GenAI tools to assist me in reviewing colleagues' code (e.g., summarizing changes or suggesting improvements). **Q27.8**: The use of GenAI tools accelerates my learning curve in new projects (onboarding), allowing me to contribute independently more quickly than usual.

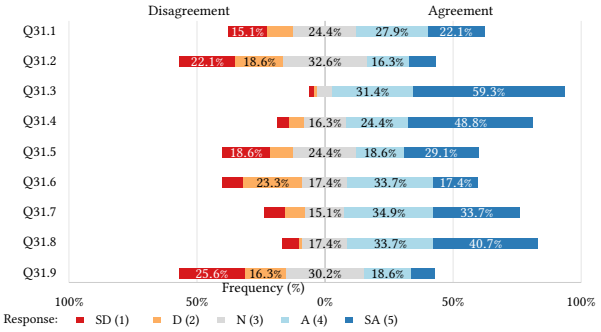
Documentation & Communication category concerns points of improvement in communication and documentation, as exemplified by Dev_54: "The use of GenAI significantly helps in the way ideas are transmitted, especially when there is difficulty in transforming mentally structured thoughts into clear and objective texts." Regarding the **Collaboration & AI Use category**, the reports reflect points about support in actions related to GenAI, as exemplified by Dev_33: "I would never replace the advice of a mentor or a more experienced colleague with GenAI; I see it as complementary support and value debate and the exchange of ideas." In the **Support for Learning & Idea Organization** grouping, as the name implies, aspects regarding support and organization are highlighted. Finally, the **Prompt Engineering** category indicates the importance of crafting a good prompt.

Finding #6. GenAI appears to support technical autonomy and onboarding acceleration (74.4%), lowering barriers to understanding legacy code (69.8%) and reducing direct dependence on experienced mentors (46.6%). However, as qualitative reports indicate that developers still value human mentor debate over AI suggestions, this individual efficiency has not yet translated into a collaborative culture of tool usage, evidenced by the low sharing of strategies and prompts among team members (53.5%).

3.3.5 Efficiency. The results are presented in Table 8. Developers generally perceive GenAI as supportive of focus and efficiency, though with some variation across tasks. About half report that GenAI helps them stay focused (Q31.1, 50.0% agreed), while others are divided or neutral. Most do not feel that prompt writing takes more time than coding itself (Q31.2, 40.7% disagreed), although some uncertainty remains. The strongest consensus appears in reducing time spent searching for technical information (Q31.3, 90.7% agreed), indicating a clear efficiency gain.

GenAI also appears to help developers overcome common workflow challenges. Many report that it assists in dealing with initial blocks or the "blank page" problem (Q31.4, 73.2% agreed) and helps reduce waiting time for assistance (Q31.7, 68.6% agreed). It also

Table 8: Distribution of responses for the *Efficiency* dimension.



Response scale: SD = Strongly disagree; D = Disagree; N = Neutral; A = Agree; SA = Strongly agree. *Items*. **Q31.1**: The use of GenAI tools helps me maintain continuous focus while programming. **Q31.2**: I often feel that I spend more time adjusting prompts (prompt engineering) to obtain the desired result than I would spend writing the code manually. **Q31.3**: GenAI has significantly reduced the time I spend searching for technical information compared with traditional search tools (e.g., Google/Stack Overflow). **Q31.4**: GenAI helps me overcome the "blank page" / initial block when starting a new task, allowing me to enter the workflow faster. **Q31.5**: The use of GenAI helps me regain work context more quickly after being interrupted (e.g., by summarizing where I stopped). **Q31.6**: I feel that the use of GenAI has reduced the cognitive effort (mental load) required to perform complex tasks, allowing me to stay focused for longer. **Q31.7**: The use of GenAI has reduced how often I get blocked on technical tasks, decreasing the waiting time I would otherwise spend awaiting help from others. **Q31.8**: The use of GenAI allows me to perform tasks outside my primary specialty (e.g., DevOps, Frontend, Testing), reducing my dependence on other teams or colleagues. **Q31.9**: The use of GenAI frequently breaks my workflow because it requires constant validation.

supports working outside one's primary area of expertise (Q31.8, 74.4% agreed). However, its impact on regaining context after interruptions is more mixed (Q31.5, 47.7% agreed), and perceptions of cognitive effort are divided (Q31.6, 51.1% agreed, 31.4% disagreed).

Finally, concerns about workflow disruption due to constant validation are not predominant, as more developers tend to disagree with this issue (Q31.9, 41.9% disagreed), although a relevant portion still perceives some impact or remains neutral.

The justifications and additional points resulted in 8 unique comments that were grouped into seven distinct categories. In the **Efficiency conditioned by the type of task grouping**, comments highlight the efficiency of AI for specific activities, as stated by Dev_54: "[...] It accelerates tasks such as organizing ideas, code review, and documentation production, reducing the time spent on repetitive or refinement activities." The **Efficiency conditioned by prompt quality** category justifies that efficiency is dependent on the quality of the prompt used, while **Efficiency in Development Support Activities** consolidates the tool as a support for general development activities. The **Cognitive Efficiency Gains** category points to gains in the cognitive dimension, whereas **Learning Limitations Associated with AI Use** considers that, in activities outside the developer's primary domain, learning may have limitations. Finally, **Efficiency Restriction by Human Factors** reports that AI use can lead to mental blocks and exhaustion and **Strategic Use of AI for Efficiency Increase** states that technology must be used strategically to achieve effective efficiency gains.

Finding #7. GenAI efficiency is conditional and strategic, showing particular value in repetitive tasks and technical refinement but can be constrained by prompt quality and the risk of mental exhaustion. Furthermore, its use in areas outside a developer's primary expertise may restrict deep learning, requiring conscious application to avoid productivity blocks.

4 Discussion

This section presents a synthesis of the main findings from the perspective of the SPACE framework dimensions. Subsequently, it conducts a complementary and exploratory quantitative analysis and compares our findings with related work.

Summary of SPACE Dimensions. We consolidated the main RQ2 findings into a conceptual image. Figure 1 presents a synthesis of the benefits and tensions reported in developers' practices of GenAI usage.

The analysis reveals that GenAI acts as a potent individual accelerator while introducing new sociotechnical challenges. In the *Satisfaction & Well-being* dimension, we observed that reducing repetitive tasks boosts confidence and daily satisfaction. However, tensions such as anxiety in specific tasks, concerns regarding technological dependency, and perceived pressure on delivery deadlines emerge. In *Performance & Quality*, the increase in delivery speed, support for code quality, and potential reduction in defects are balanced by the indispensable need for human review due to the risk of subtle bugs and security concerns.

In the *Activity* dimension, GenAI support is evident in development tasks, debugging, code explanation, and documentation. Nonetheless, it is less frequently used in management-related activities. For *Communication & Collaboration*, the findings point to improvements in technical understanding, onboarding, and communication clarity. Conversely, prompt sharing is not yet a standardized practice, and AI-generated PRs raise concerns about subtle logical errors. Finally, regarding the *Efficiency* dimension, GenAI enables faster information retrieval, supports activities outside the developer's primary domain, and reduces technical blocks. However, further improvements are needed to mitigate the time spent on prompt engineering, workflow disruptions, and constant validation overhead. We can thus emphasize that while the use of GenAI tools leads to noticeable improvements in perceived developer productivity, these gains are conditioned upon human supervision, continuous validation, and responsible use.

Supplementary Statistical Analysis. To complement our findings, we conducted exploratory stratified comparisons based on seniority and GenAI usage frequency. Participants were grouped by seniority into early-career developers, including interns, trainees, and juniors ($n = 20$), mid-level developers ($n = 30$), and senior developers ($n = 36$). They were also grouped by usage frequency into intensive users, who reported using GenAI tools more than once a day ($n = 50$), and other users ($n = 36$). Before computing the composite scores used in these comparisons, we assessed the internal consistency of each item grouping using Cronbach's alpha and retained only groupings with $\alpha \geq 0.60$ [19].

Items were initially classified as benefits or tensions according to the semantic orientation of each statement. Benefit scores aggregate Likert items describing desirable effects of GenAI use, such as improved satisfaction, quality, collaboration, or efficiency (Q15.1, Q15.7–Q15.9, Q19.4, Q19.7-Q19.8, Q27.1, Q27.3–Q27.8, Q31.1, Q31.3–Q31.8). Tension scores aggregate Likert items describing perceived costs or risks, such as anxiety, dependency, pressure, bugs, security concerns, or validation overhead (Q15.2–Q15.6, Q19.1–Q19.3, Q19.5-Q19.6, Q27.2). The *Activity* dimension was analyzed separately as a frequency score based on Q23.1–Q23.9. Items Q31.2 and

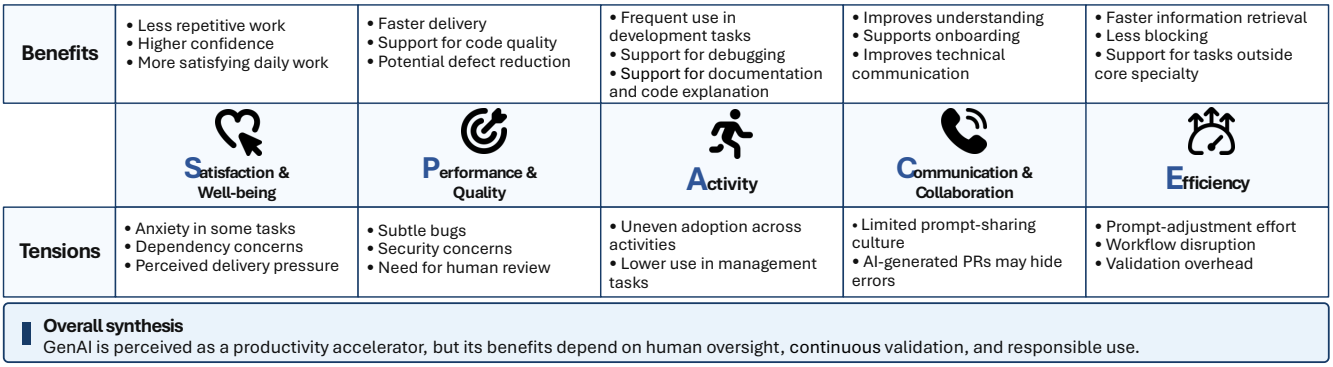


Figure 1: Synthesis of GenAI Benefits and Tensions across the SPACE Framework Dimensions

Table 9: Exploratory stratified comparisons of composite scores derived from SPACE items.

Outcome	Group medians [IQR]	p	p_{Holm}	Effect size
Seniority comparison				
Satisfaction benefits	E: 3.75 [3.25–4.06]; M: 3.88 [3.25–4.25]; S: 3.25 [2.69–3.56]	0.006	0.056	ϵ^2 = 0.101, moderate
Satisfaction tensions	E: 2.70 [2.35–3.25]; M: 2.80 [2.20–3.40]; S: 2.60 [2.20–3.40]	0.873	1.000	ϵ^2 = 0.000, negligible
Performance benefits	E: 3.67 [3.25–4.33]; M: 3.83 [3.00–4.33]; S: 3.67 [3.00–4.00]	0.608	1.000	ϵ^2 = 0.000, negligible
Performance tensions	E: 3.50 [3.15–4.05]; M: 3.40 [3.20–4.00]; S: 3.40 [3.15–3.80]	0.654	1.000	ϵ^2 = 0.000, negligible
Activity frequency	E: 3.56 [3.22–4.00]; M: 3.78 [3.06–4.31]; S: 3.56 [3.19–3.92]	0.608	1.000	ϵ^2 = 0.000, negligible
Communication benefits	E: 3.71 [3.29–4.14]; M: 3.93 [3.14–4.39]; S: 3.43 [2.57–3.75]	0.033	0.261	ϵ^2 = 0.058, small
Communication tensions	E: 3.00 [1.75–4.00]; M: 3.00 [2.00–4.00]; S: 3.00 [3.00–4.00]	0.631	1.000	ϵ^2 = 0.000, negligible
Efficiency benefits	E: 4.00 [3.79–4.61]; M: 4.14 [3.61–4.43]; S: 3.57 [3.00–4.00]	0.009	0.084	ϵ^2 = 0.089, moderate
Efficiency prompt effort	E: 3.00 [2.00–3.00]; M: 3.00 [2.00–4.00]; S: 2.50 [1.00–3.00]	0.087	0.608	ϵ^2 = 0.035, small
Efficiency validation	E: 3.00 [1.75–4.00]; M: 3.00 [2.00–4.00]; S: 2.50 [1.00–3.00]	0.089	0.608	ϵ^2 = 0.034, small
Usage intensity comparison				
Satisfaction benefits	I: 3.62 [3.25–4.25]; O: 3.25 [2.44–3.75]	0.016	0.096	δ = 0.304, small
Satisfaction tensions	I: 2.60 [2.20–3.40]; O: 2.80 [2.20–3.40]	0.626	1.000	δ = -0.062, negligible
Performance benefits	I: 3.83 [3.08–4.33]; O: 3.33 [2.67–4.00]	0.013	0.093	δ = 0.312, small
Performance tensions	I: 3.20 [3.00–3.95]; O: 3.70 [3.40–4.00]	0.022	0.111	δ = -0.289, small
Activity frequency	I: 3.89 [3.56–4.44]; O: 3.17 [2.75–3.56]	<.001	<.001	δ = 0.626, large
Communication benefits	I: 3.86 [3.32–4.29]; O: 3.21 [2.43–3.71]	<.001	0.004	δ = 0.449, medium
Communication tensions	I: 3.00 [2.00–4.00]; O: 3.00 [3.00–4.00]	0.762	1.000	δ = -0.037, negligible
Efficiency benefits	I: 4.07 [3.57–4.43]; O: 3.64 [2.57–4.04]	0.002	0.016	δ = 0.392, medium
Efficiency prompt effort	I: 3.00 [1.25–3.00]; O: 3.00 [2.00–4.00]	0.479	1.000	δ = -0.088, negligible
Efficiency validation	I: 3.00 [1.00–3.00]; O: 3.00 [2.00–4.00]	0.425	1.000	δ = -0.099, negligible

Notes. E = early-career developers, including interns, trainees, and juniors; M = mid-level developers; S = senior developers. I = intensive users, who reported using GenAI more than once a day; O = other users. p_{Holm} denotes Holm-adjusted p-values. Benefit- and tension-oriented composite scores were computed by grouping items according to the semantic orientation of each statement and retained when their internal consistency reached $\alpha \geq 0.60$. Q31.2 and Q31.9 were analyzed separately as efficiency-related single-item outcomes. Results are exploratory; Likert and frequency scores range from 1 to 5.

Q31.9 were not aggregated into a composite score because their grouping showed internal consistency below the adopted threshold ($\alpha < 0.60$). Only Q19.3 was reverse-coded so that higher scores consistently represented stronger performance-related tensions.

Since the scores were derived from Likert and frequency scales, we used non-parametric tests: Kruskal-Wallis for seniority-based comparisons and Mann-Whitney U for usage-frequency comparisons. We adjusted p-values using the Holm correction and reported effect sizes using epsilon-squared and Cliff's delta, respectively. When applicable, Dunn's post-hoc test with Holm correction was used for pairwise comparisons. The complete analysis is available in our replication package [10]. Table 9 summarizes the results of the exploratory stratified comparisons.

We found that the most consistent differences were associated with GenAI usage intensity. After Holm correction, intensive users reported higher *Activity frequency*, *Communication benefits*, and *Efficiency benefits*, with medium to large effect sizes. This suggests that developers who use GenAI more than once a day tend to integrate these tools more broadly into daily development activities and perceive stronger support for onboarding, technical understanding, information retrieval, and reduced blocking.

Seniority-based differences were more limited. Although early-career and mid-level developers tended to report higher *Satisfaction benefits* and *Efficiency benefits* than senior developers, these differences did not remain significant after Holm correction. Thus, the results suggest that usage intensity is more clearly associated with perceived GenAI benefits than seniority. Tension-related outcomes, including the single-item efficiency outcomes related to prompt effort and validation overhead, did not consistently increase among intensive users, which may indicate that frequent users develop better strategies for validating and integrating GenAI outputs.

5 Related Work

Drucker [14] defines knowledge worker productivity through quality and task management autonomy. In SE, the concept is complex and not limited to isolated metrics [23]. As knowledge-intensive work, productivity results from collaboration and the exchange of ideas among developers [13, 17, 28, 40].

GenAI is considered a transformative technology that automates routine tasks and assists in implementation, debugging, and documentation [20, 21, 34, 43]. Initial studies indicate positive correlations between the use of these tools and individual productivity, focusing on reducing the execution time of technical activities [11, 12, 29]. However, impacts vary according to task complexity and the developer's experience [5, 15]. The literature emphasizes that AI-driven efficiency must be balanced with human oversight to avoid overreliance and the degradation of technical rigor [4, 30, 38].

Recent studies have applied the SPACE framework and related productivity models to investigate the impact of GenAI tools on software development. Peng et al. [35] conducted the first controlled experiment to measure the impact of GitHub Copilot on productivity. They recruited 95 programmers, dividing them into treatment and control groups, and assigned them the task of implementing an HTTP server in JavaScript. The group with access to Copilot completed the task 55.8% faster. Our results converge, as the most

cited benefits associated with GenAI tools were Productivity (69.8%) and Technical Understanding & Learning (41.9%).

Bird et al. [6] examined the early experiences of GitHub Copilot in three studies: a forum analysis, a think-aloud study, and a survey. They observed that developers spent less time on Stack Overflow but also exhibited a lower understanding of how the generated code worked. They also found that users spent more time reading and reviewing code than writing it. Our findings extend this discussion by showing that developers reported concerns regarding technological dependency and skill atrophy (57.0%).

Wivestad and Ulfsnes [47] found strong associations between GenAI and the Satisfaction and Efficiency dimensions of the SPACE framework, but weaker relationships with Performance and Communication & Collaboration. Our findings are consistent with these results and provide a possible explanation for them. Developers reported substantial gains in satisfaction due to the reduction of repetitive tasks (83.7%). However, faster delivery of valuable features (77.9%) coexisted with the occurrence of subtle bugs (58.1%) and a strong need for human review (86.0%), suggesting that productivity gains are partially offset by the effort required for validation.

Our results also complement the study by Ziegler et al. [49], which identified the acceptance rate as the primary predictor of perceived productivity. While their findings suggest that useful AI suggestions drive productivity gains, our results indicate that acceptance alone does not fully capture productivity, as developers still devote considerable effort to reviewing AI-generated outputs.

Finally, Coelho et al. [11] reported anticipated benefits and drawbacks of GenAI among Brazilian software developers. Our study complements these findings by providing prevalence data, including benefits for onboarding new team members (74.4%), limited prompt sharing among colleagues (53.5%), and the occurrence of subtle bugs in AI-generated code (58.1%).

6 Trustworthiness of the Study

Regarding **construct threats**, the research questions were grounded in the literature on productivity and GenAI [11, 15]. To mitigate construct misinterpretation, the survey followed the SPACE framework, was reviewed by experienced SE researchers, and was piloted with developers from different seniority levels. As for **internal threats**, the study relies on self-reported perceptions, which may be affected by recall bias, social desirability bias, and prior attitudes toward GenAI. Therefore, our findings reflect perceived impacts rather than objective performance measures. Regarding **conclusion threats**, the findings may not fully support all possible inferences drawn from the data. To reduce confirmation and interpretation bias, quantitative and qualitative results were discussed among the researchers. The qualitative analysis followed inductive open coding with multiple researchers, calibration meetings, consensus discussions, thematic classification, and final review.

Finally, non-probabilistic convenience sampling limits **external validity**. Although the final sample ($N = 86$) is adequate for an exploratory analysis, it may not represent the broader software industry. In particular, the sample is predominantly composed of developers working in large companies and professionals with more than six years of experience, which may limit the generalizability of the findings.

Although the sample provided quantitative and qualitative evidence across all five SPACE dimensions, it may not capture the full range of developers' experiences within each dimension. Thus, this study offers initial empirical evidence rather than definitive generalizations.

7 Conclusion

This study investigates the impact of GenAI tools on developer productivity through the lens of the five SPACE framework dimensions. The main contribution of this work lies in the multidimensional synthesis of GenAI benefits and tensions within contemporary industrial practice. Beyond the technical analysis per dimension, we provide an open repository containing the categorized developer responses. This initiative aims to strengthen scientific research integrity by enabling auditability, transparency, and data reuse by other researchers for comparative studies or meta-analyses.

Our study points to several practical implications. Quality concerns hold implications for GenAI adoption policies. Given the frequency of subtle bugs, teams should classify these tools as drafting assistants rather than autonomous generators. Organizations must update code review guidelines to mandate human validation. Developers can benefit from these findings by using the SPACE dimensions to reflect on and adjust their workflows, particularly by encouraging the sharing of GenAI strategies and best practices, since many currently do not share them (53.5%), which may lead to knowledge silos. At the organizational level, companies can use these insights to improve governance and support practices, especially by addressing reported pressure for faster delivery and ensuring that GenAI adoption does not negatively impact well-being. The observed fear of cognitive loss (57.0%) also suggests the need for continuous training to reinforce programming fundamentals.

As future work, we intend to extend this investigation through the following avenues: (i) Exhaustive Qualitative Analysis, processing the full volume of open-ended responses that could not be detailed here due to page limits; (ii) Deep Statistical Analysis, conducting correlation studies between seniority, domain, and usage frequency with quality and satisfaction perceptions; and (iii) Experimental Validation, performing a controlled experiment to compare the practical performance of developers with and without GenAI assistance, confronting metric results with the perceptions reported in this survey.

Artifact Availability

The artifacts supporting this study are available in the replication package repository: https://doi.org/10.5281/zenodo.22135084. The repository contains anonymized survey data, qualitative coding, quantitative datasets, and statistical analysis outputs used in this study. All participant-related data were sanitized prior to disclosure. The artifacts will remain publicly available after publication.

Acknowledgements

This work was supported in part by the Atlântico Institute, the State University of Ceará (UECE), and National Council for Scientific and Technological Development (CNPq - PDI 120070/2025-1). We used LLMs for language refinement, but all analysis and interpretation remain our sole responsibility.

References

[1] Aldeida Aleti. 2023. Software testing of generative ai systems: Challenges and opportunities. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE, 4–14.

[2] Anthropic. 2026. *Claude*. https://claude.ai Large language model. Accessed: 2026-04-29.

[3] Sebastian Baltes and Paul Ralph. 2022. Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering* 27, 4 (2022), 94.

[4] Leonardo Banh, Florian Holldack, and Gero Strobel. 2025. Copiloting the future: How generative AI transforms Software Engineering. *Information and Software Technology* 183 (2025), 107751.

[5] Brett A Becker, Michelle Craig, Paul Denny, Hieke Keuning, Natalie Kiesler, Juho Leinonen, Andrew Luxton-Reilly, James Prather, and Keith Quille. 2023. Generative AI in introductory programming. *Computer Science Curricula* (2023), 438–439.

[6] Christian Bird, Denae Ford, Thomas Zimmermann, Nicole Forsgren, Eirini Kalliamvakou, Travis Lowdermilk, and Idan Gazit. 2023. Taking Flight with Copilot. *Commun. ACM* 66, 6 (May 2023), 56–62. https://doi.org/10.1145/3589996

[7] Edna Dias Canedo and Giovanni Almeida Santos. 2019. Factors affecting software development productivity: An empirical study. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. 307–316.

[8] Valerie Chen, Jasmyn He, Behnjamin Williams, Jason Valentino, and Ameet Talwalkar. 2026. Beyond the Commit: Developer Perspectives on Productivity with AI Coding Assistants. *arXiv preprint arXiv:2602.03593* (2026).

[9] Haowei Cheng, Jati H Husen, Yijun Lu, Teeradaj Racharak, Nobukazu Yoshioka, Naoyasu Ubayashi, and Hironori Washizaki. 2026. Generative ai for requirements engineering: A systematic literature review. *Software: Practice and Experience* 56, 2 (2026), 141–170.

[10] Murilo Coelho, Francisco Dione de Sousa Amâncio, Matheus Paixao, and Edson S. S. Freire. 2026. Replication Package for the paper: More Productive, but at What Cost? Understanding How GenAI Shapes Developers' Work Across the SPACE Dimensions. https://doi.org/10.5281/zenodo.21271232. https://doi.org/10.5281/zenodo.21271233 Dataset for SBES 2026.

[11] Murilo Coelho, Isabelle Reinbold, Lizie Sancho, Matheus Paixao, Allysson Allex Araújo, and Sávio Freire. 2025. Software Developers' Perceptions of Productivity: An Industry-focused Study. In *Simpósio Brasileiro de Qualidade de Software (SBQS)*. SBC, 12–22.

[12] Mariana Coutinho, Lorena Marques, Anderson Santos, Marcio Dahia, Cesar França, and Ronnie de Souza Santos. 2024. The role of generative ai in software development productivity: A pilot case study. In *Proceedings of the 1st ACM International Conference on AI-Powered Software*. 131–138.

[13] Oscar Dieste, Alejandrina Aranda, Fernando Uyaguari, Burak Turhan, Ayse Tosun, Davide Fucci, Markku Oivo, and Natalia Juristo. 2018. Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study. In *Proceedings of the 2018 International Conference on Software and System Process*. 111–112.

[14] P Drucker. 2001. Knowledge worker productivity. *California Management Review* 41, 2 (2001), 45–49.

[15] Samuel Ferino, Rashina Hoda, John Grundy, and Christoph Treude. 2025. Novice developers' perspectives on adopting llms for software development: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* (2025).

[16] Denae Ford, Margaret-Anne Storey, Thomas Zimmermann, Christian Bird, Sonia Jaffe, Chandra Maddila, Jenna L Butler, Brian Houck, and Nachiappan Nagappan. 2021. A tale of two cities: Software developers working from home during the covid-19 pandemic. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 2 (2021), 1–37.

[17] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. 2021. The SPACE of Developer Productivity. *ACM Queue* 19, 1 (2021). January–February issue.

[18] GitHub. 2026. *GitHub Copilot*. https://github.com/features/copilot Accessed: 2026-04-29.

[19] JF Hair, BJ Babin, RE Anderson, and WC Black. 2019. Multivariate Data Analysis. England Pearson Prentice. *References-Scientific Research Publishing.(Nd)* (2019).

[20] Ahmed E Hassan, Dayi Lin, Gopi Krishnan Rajbahadur, Keheliya Gallaba, Filipe Roseiro Cogo, Boyuan Chen, Haoxiang Zhang, Kishanthan Thangarajah, Gustavo Oliva, Jiahuei Lin, et al. 2024. Rethinking software engineering in the era of foundation models: A curated catalogue of challenges in the development of trustworthy fmware. In *Companion Proceedings of the 32nd ACM international conference on the foundations of software engineering*. 294–305.

[21] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024), 1–79.

[22] Sajed Jalil. 2025. The transformative influence of llms on software development & developer productivity. In *2025 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*. IEEE, 1–10.

[23] Ciera Jaspan and Caitlin Sadowski. 2019. No Single Metric Captures Productivity. (2019).

[24] Peter Kokol. 2024. The use of AI in software engineering: A synthetic knowledge synthesis of the recent research literature. *Information* 15, 6 (2024), 354.

[25] Sanwal Manish and Deva Ishan. 2024. A Multi-Faceted Approach to Measuring Engineering Productivity. *International Journal of Trend in Scientific Research and Development* 8, 5 (2024), 516–521.

[26] João José Maranhão and Eduardo Martins Guerra. 2024. A prompt pattern sequence approach to apply generative AI in assisting software architecture decision-making. In *Proceedings of the 29th European Conference on Pattern Languages of Programs, People, and Practices*. 1–12.

[27] Philipp Mayring. 2021. Qualitative content analysis: A step-by-step guide. (2021).

[28] André N Meyer, Thomas Fritz, Gail C Murphy, and Thomas Zimmermann. 2014. Software developers' perceptions of productivity. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*. 19–29.

[29] Ekaterina A Moroz, Vladimir O Grizkevich, and Igor M Novozhilov. 2022. The potential of artificial intelligence as a method of software developer's productivity improvement. In *2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*. IEEE, 386–390.

[30] Anh Nguyen-Duc, Beatriz Cabrero-Daniel, Adam Przybylek, Chetan Arora, Dron Khanna, Tomas Herda, Usman Rafiq, Jorge Melegati, Eduardo Guerra, Kai-Kristian Kemell, et al. 2025. Generative artificial intelligence for software engineering—A research agenda. *Software: Practice and Experience* 55, 11 (2025), 1806–1843.

[31] Abi Noda, Margaret-Anne Storey, Nicole Forsgren, and Michaela Greiler. 2023. Devex: What actually drives productivity? *Commun. ACM* 66, 11 (2023), 44–49.

[32] Edson Oliveira, Davi Viana, Marco Cristo, and Tayana Conte. 2017. How have software engineering researchers been measuring software productivity?-a systematic mapping study. In *International conference on enterprise information systems*, Vol. 2. SciTePress, 76–87.

[33] OpenAI. 2026. *ChatGPT*. https://chat.openai.com Large language model. Accessed: 2026-04-29.

[34] Ipek Ozkaya. 2023. Application of large language models to software engineering tasks: Opportunities, risks, and implications. *IEEE software* 40, 3 (2023), 4–8.

[35] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590* (2023).

[36] Crystal Qian and James Wexler. 2024. Take it, leave it, or fix it: Measuring productivity and trust in human-ai collaboration. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. 370–384.

[37] Asha Rajbhoj, Akanksha Somase, Piyush Kulkarni, and Vinay Kulkarni. 2024. Accelerating software development using generative AI: ChatGPT case study. In *Proceedings of the 17th innovations in software engineering conference*. 1–11.

[38] Sanka Rasnayaka, Guanlin Wang, Ridwan Shariffdeen, and Ganesh Neelakanta Iyer. 2024. An empirical study on usage and perceptions of llms in a software engineering project. In *Proceedings of the 1st international workshop on large language models for code*. 111–118.

[39] Patricia de Oliveira Santos, Allan Chamon Figueiredo, Pedro Nuno Moura, Bruna Diirr, Adriana CF Alvim, and Rodrigo Pereira Dos Santos. 2024. Impacts of the usage of generative artificial intelligence on software development process. In *Proceedings of the 20th Brazilian Symposium on Information Systems*. 1–9.

[40] Walt Scacchi and D Hurley. 1995. Understanding software productivity. *Software Engineering and Knowledge Engineering: Trends for the next decade* 4 (1995), 273–316.

[41] Margaret-Anne Storey, Thomas Zimmermann, Christian Bird, Jacek Czerwonka, Brendan Murphy, and Eirini Kalliamvakou. 2019. Towards a theory of software developer job satisfaction and perceived productivity. *IEEE Transactions on Software Engineering* 47, 10 (2019), 2125–2142.

[42] Anselm Strauss, Juliet Corbin, et al. 1990. *Basics of qualitative research*. Vol. 15. sage Newbury Park, CA.

[43] Runchu Tian, Yining Ye, Yujia Qin, Xin Cong, Yankai Lin, Yinxu Pan, Yesai Wu, Hui Haotian, Liu Weichuan, Zhiyuan Liu, et al. 2024. Debugbench: Evaluating debugging capability of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*. 4173–4198.

[44] Iman Tunggono and Elfindah Princes. 2025. Understanding Developer Productivity: Input-Output Perspectives Within the SPACE Framework. In *2025 International Conference on Computing and Applied Informatics (ICCAI)*. IEEE, 1–7.

[45] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*. 1–7.

[46] Justin D Weisz, Shraddha Vijay Kumar, Michael Muller, Karen-Ellen Browne, Arielle Goldberg, Katrin Ellice Heintze, and Shagun Bajpai. 2025. Examining the use and impact of an ai code assistant on developer productivity and experience in the enterprise. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–13.

[47] Viggo Tellefsen Wivestad and Rasmus Ulfsnes. 2024. A Journey Through SPACE: Unpacking the Perceived Productivity of GitHub Copilot. In *International Conference on Agile Software Development*. Springer, 42–50.

[48] Claes Wohlin, Per Runeson, Martin Host, Magnus C. Ohlsson, Bjorn Regnell, and Anders Wesslen. 2012. *Experimentation in Software Engineering*. Springer.
[49] Albert Ziegler, Eirini Kalliamvakou, X Alice Li, Andrew Rice, Devon Rifkin, Shawn Simister, Ganesh Sittampalam, and Edward Aftandilian. 2024. Measuring github copilot's impact on productivity. *Commun. ACM* 67, 3 (2024), 54–63.
[50] Thomas Zimmermann. 2022. Measuring Developer Productivity and the New Future of Work. In *Proceedings of the 15th Innovations in Software Engineering Conference*. 1–1.