Are We Measuring the Wrong Things? How Software Project Leaders View Gen-AI-Assisted Developer Productivity

Ana Carolina Chagas
Institute of Computing/Federal University of Amazonas
Manaus, Brazil
carolinachagas@icomp.ufam.edu.br

Rayfran Rocha Lima
Sidia Institute of Science and Technology
Manaus, Brazil
rayfran.rocha.lima@gmail.com

Ana Carolina Oran
Institute of Computing/Federal University of Amazonas
Manaus, Brazil
ana.oran@icomp.ufam.edu.br

Abstract

The adoption of Generative Artificial Intelligence (Gen-AI) in software development has rapidly expanded across activities such as code generation, debugging, documentation, requirements analysis, and test creation. Although previous studies frequently report productivity gains associated with faster execution and increased delivery speed, less attention has been given to how Gen-AI reshapes the interpretation of productivity, particularly from the perspective of those responsible for evaluating developers. Traditional productivity indicators, including lines of code, commits, and delivery speed, were designed for workflows centered on direct artifact production and may become less reliable proxies for interpreting developer contribution in Gen-AI-assisted environments. This study investigates how software project leaders perceive developer productivity in contexts where Gen-AI tools are integrated into everyday software development practices. We conducted a survey with 20 professionals involved in managing, evaluating, or leading software development teams. The study combined quantitative and qualitative analysis to investigate perceived productivity changes, evaluation criteria, cognitive and organizational impacts, and distinctions between human contribution and Gen-AI-assisted productivity. The results suggest an emerging process of productivity reinterpretation in which productivity becomes increasingly associated with validation effort, interpretation, technical judgment, and effective use of Gen-AI tools rather than only observable code-based artifacts. Participants reported less emphasis on code volume and stronger attention to autonomy, problem understanding, solution quality, and critical assessment of generated outputs. Dependency, learning dynamics, and fairness in performance evaluation also emerged as relevant concerns when distinguishing human contribution from tool-assisted output. Our findings suggest that Gen-AI not only accelerates software development activities, but also reshapes how leaders interpret and assess developer productivity. By examining productivity from the perspective of software project leaders, this study contributes to ongoing discussions about the limitations of traditional productivity signals in increasingly Gen-AI-assisted software engineering environments.

Keywords

Productivity, Generative Artificial Intelligence, Software Project Leadership, Empirical Study

1 Introduction

The adoption of Generative Artificial Intelligence (Gen-AI) in software development has expanded rapidly in recent years, influencing activities such as code generation, debugging, documentation, requirements engineering, and test creation [14]. Rather than acting only as auxiliary support, these tools have become integrated into everyday development workflows, changing how software tasks are executed and how technical effort is distributed. As developers increasingly alternate between implementation activities and interactions involving prompts, iterative refinement, and evaluation of generated outputs, part of the development effort shifts from direct artifact production toward interpretation, validation, and technical decision-making.

This transformation challenges traditional interpretations of software productivity. Conventional productivity indicators such as lines of code, commits, and delivery speed were designed for workflows centered on direct artifact production [6]. In Gen-AI-assisted environments, however, substantial effort is devoted to validating generated solutions, adapting outputs to project contexts, refining prompts, and assessing technical correctness [15]. Consequently, observable software artifacts may no longer fully represent developer contribution.

Although previous studies report productivity gains associated with Gen-AI adoption, most investigations remain focused on operational efficiency or developer perception [11]. Less attention has been given to how software project leaders reinterpret productivity and evaluate developer contribution as Gen-AI becomes part of everyday development practices. Since productivity assessment depends not only on measurable outputs but also on managerial interpretation of quality, autonomy, problem-solving ability, and technical judgment, Gen-AI may transform how contribution and merit are recognized within development teams [12].

In this paper, we investigate how software project leaders perceive developer productivity in contexts where Gen-AI tools are integrated into software development workflows. We conducted a survey with 20 professionals involved in evaluating, managing, or leading software development teams. The study examines perceived changes in productivity, current evaluation criteria, and expectations regarding developer performance in Gen-AI-assisted environments.

The study is guided by the following research question: *How is the adoption of Gen-AI tools modifying the notion of productivity according to software project leaders' perceptions?* This study contributes to ongoing discussions on software productivity in Gen-AI-assisted environments by providing exploratory empirical insights into how productivity assessment is becoming increasingly interpretation-oriented rather than exclusively artifact-centered.

The remainder of this paper is organized as follows. Section 2 presents the background and related work. Section 3 describes

the research method. Section 4 presents and discusses the study results. Section 5 discusses the threats to validity. Finally, Section 6 concludes the paper and outlines directions for future work.

2 Background and Related Work

This section introduces the background necessary to understand how Gen-AI is reshaping software development activities and productivity evaluation. It also discusses existing studies on developer productivity in Gen-AI-assisted environments and positions the present work within the current literature.

2.1 Gen-AI-assisted Software Development

Recent advances in Generative Artificial Intelligence (Gen-AI) have expanded the use of Gen-AI tools in software development activities, including code generation, debugging, documentation, requirements analysis, and information retrieval [16]. These tools are increasingly integrated into everyday engineering practices through continuous interaction between developers and generated outputs, primarily mediated by natural language prompts that developers iteratively refine according to tool responses [17].

As Gen-AI becomes part of software development workflows, developers increasingly alternate between requesting draft solutions, reviewing generated suggestions, and deciding whether generated outputs should be accepted, modified, or discarded. Although these tools accelerate information retrieval and initial solution drafting, recurring limitations such as hallucinated information, inaccurate outputs, and incomplete contextual understanding require continuous human verification and technical judgment [7]. Consequently, part of software development effort is redistributed toward validation, interpretation, refinement, and technical assessment of generated outputs rather than direct implementation activities [9].

This redistribution challenges traditional interpretations of software engineering productivity. Historically, productivity has been inferred from observable artifacts such as lines of code, commits, pull requests, and delivery speed [6]. In Gen-AI-assisted workflows, however, these artifacts may no longer fully represent developer contribution because substantial effort is devoted to validating generated solutions, assessing alignment with business requirements, refining prompts, and evaluating technical correctness before accepting AI-generated outputs.

2.2 Productivity Evaluation in Gen-AI-assisted Environments

Developer evaluation has traditionally combined quantitative indicators with qualitative dimensions such as autonomy, collaboration, communication, problem-solving ability, and technical decision-making [5]. In Gen-AI-assisted environments, these qualitative dimensions become increasingly important because productivity depends not only on what developers produce, but also on how they interpret, validate, and operationalize generated outputs [11]. This perspective is particularly relevant for software project leaders, who are responsible for interpreting productivity signals, evaluating developer contribution, and making decisions related to recognition, promotion, and team performance management [1].

Recent research has investigated Gen-AI adoption in software engineering from different perspectives, including usage patterns, perceived productivity, workflow acceleration, and developer experience. Existing studies report frequent use of Gen-AI for code generation, debugging, documentation support, requirements clarification, and information retrieval, while also highlighting recurring concerns involving inaccurate outputs, hallucinations, incomplete requirement understanding, and dependency risks [2]. These limitations require continuous human supervision and technical judgment [7], suggesting that the effects of Gen-AI extend beyond operational acceleration and involve changes in how software work is interpreted and validated.

Some recent studies have expanded this discussion by adopting human-centered perspectives on productivity [10], examining the redistribution of developer effort toward verification, oversight, and other cognitively demanding activities [4], and discussing managerial challenges associated with Gen-AI adoption [13]. Despite these advances, many approaches to assessing developer productivity still rely on self-reported productivity perception, operational efficiency gains, or traditional repository-based metrics such as commits and delivery speed [11]. As a result, they may not fully capture the cognitive effort associated with interpreting, validating, and refining Gen-AI-generated outputs. This limitation becomes more evident when generated artifacts partially obscure the distinction between human contribution and automated assistance. Recent work also suggests that heavy reliance on Gen-AI tools may affect how expertise, ownership, and technical contribution are inferred from repository activity and authored code [14].

Existing studies have advanced the understanding of Gen-AI adoption by investigating productivity gains, developer experiences, human-centered productivity dimensions, workflow changes, repository-based productivity indicators, and managerial challenges. However, they provide more limited insights into how software project leaders reinterpret productivity when evaluating developer contribution in Gen-AI-assisted environments. Rather than proposing new productivity metrics or examining productivity from the perspective of developers, our study investigates how those responsible for evaluating software professionals understand productivity in AI-assisted software development. By focusing on software project leaders' evaluation criteria and managerial interpretation of developer contribution, this work complements existing research by addressing a managerial perspective that remains comparatively less explored in the current literature.

3 Research Method

To investigate how the adoption of Gen-AI tools modifies the notion of productivity according to software project leaders' perceptions, we conducted a survey study. Survey research is commonly used to capture opinions and perceptions in a structured manner, enabling the collection of comparable data across participants [8].

The study design followed the main stages of questionnaire-based research discussed by Kitchenham and Pfleeger [8], including definition of research objectives, questionnaire design, instrument evaluation, data collection, and data analysis.

The **research objectives** were defined based on gaps identified in the literature regarding how software project leaders perceive developer productivity in Gen-AI-assisted environments. During the design stage, the target population, questionnaire items, and

data collection strategy were defined. A pilot study was conducted to refine the questionnaire before its application.

Questionnaire Design. The target population consisted of professionals in leadership roles responsible for evaluating software developers whose teams use Gen-AI tools in software development activities. The questionnaire was designed to investigate perceptions related to productivity in Gen-AI-assisted environments, including usage context, perceived productivity changes, cognitive and organizational impacts, evaluation practices, productivity metrics, and distinctions between human and Gen-AI-assisted productivity.

The instrument was implemented using Google Forms and contained 49 questions distributed across thematic sections, as summarized in Table 1. The questionnaire combined Likert-scale, multiple-choice, and open-ended questions, enabling the collection of both quantitative and qualitative data. Before accessing the questionnaire, participants received a consent form explaining the study objectives, anonymity conditions, and voluntary participation. The complete questionnaire is available in the supplementary material repository.

Table 1: Questionnaire structure and question types

Section	Question examples	Type
Participant Profile	Leadership role, software experience, organization type, work model	Multiple-choice
Gen-AI Usage Context	Frequency of use, adopted tools, supported development activities	Multiple-choice / Likert
Perceived Productivity Changes	Impact on delivery speed, code quality, rework, documentation, learning, and effort	Likert / Open-ended
Cognitive Impacts	Effects on focus, autonomy, confidence, learning ability, and motivation	Likert / Open-ended
Organizational Impacts	Effects on communication, mentoring, collaboration, and knowledge sharing	Likert / Open-ended
Developer Productivity Attributes	Importance of technical knowledge, collaboration, proactivity, critical thinking, and problem solving before and after Gen-AI adoption	Likert / Open-ended
Human vs AI-Assisted Productivity	Differences between human and AI-assisted productivity	Open-ended
Evaluation Metrics and Performance Assessment	Metrics used to evaluate productivity and Gen-AI impact on evaluation criteria	Multiple-choice / Open-ended

Instrument evaluation. The questionnaire was reviewed by two researchers and evaluated through a pilot study with one participant, a team leader with more than 15 years of experience in a large organization. The pilot study enabled the identification of issues related to question clarity and structure. Based on this feedback, minor adjustments were made to improve the instrument

Data Collection. The questionnaire was distributed through LinkedIn, email invitations, and personal contacts between January and February 2026. The average completion time was approximately 15 minutes. A total of 22 responses were collected. Two responses were excluded from the analysis, one corresponding to the pilot study and another due to corrupted data. The final dataset consisted of 20 valid responses. All responses were collected anonymously, and no sensitive data were requested.

Data Analysis. The collected data were analyzed using quantitative and qualitative approaches. For closed-ended questions, statistical analysis was performed using Python to identify distributions and general trends in participants' responses. For open-ended questions, a thematic analysis [3] was conducted using Atlas.ti. The analysis considered responses to 20 open-ended questions provided by the 20 participants, resulting in a dataset comprising 68 codes and 412 coded quotations. The coding process involved identifying recurring concepts through iterative refinement of the initial codes. The resulting categories were consolidated based on the research subquestions defined during the study design, supporting the interpretation of findings in relation to the study objectives. The coding was also reviewed by a second researcher to improve consistency throughout the analysis.

4 Results and Discussion

This section presents the results of the study, including demographic data, quantitative results, and qualitative insights derived from participants' responses.

4.1 Demographic Data

This study included 20 professionals involved in software development leadership activities in teams that use Generative Artificial Intelligence (Gen-AI) tools during software development processes. Participants occupied different leadership positions, including Project Leaders (40%), Technical Leaders (25%), QA Leaders (10%), Technical Coordinators (10%), Project Managers (10%), and Engineering Managers (5%). These roles reflect responsibilities associated with technical decision-making, team coordination, and delivery management.

Participants also reported acting in multiple software engineering domains, including product-related activities (45%), software development (40%), quality engineering (15%), software architecture (15%), DevOps/SRE/infrastructure (15%), machine learning (10%), governance/PMO (10%), and information security (5%). This distribution indicates that the study includes leaders operating across both technical and coordination-oriented contexts.

Regarding professional experience in software development, participants reported the following ranges: less than 2 years (10%), between 2 and 5 years (30%), between 6 and 10 years (5%), between 11 and 15 years (5%), and more than 15 years (50%). A larger proportion of participants therefore reported senior experience levels, particularly above 15 years.

Participants also reported working in startups (20%), medium-sized companies (30%), large enterprises (35%), and public institutions (15%). Different working models were also represented, including fully remote (40%), hybrid (45%), and fully on-site arrangements (15%). This diversity provides an exploratory view of how productivity is interpreted across different organizational and leadership contexts in Gen-AI-assisted software development environments.

4.2 Quantitative Results

This section presents the quantitative results regarding perceived changes in productivity, technical aspects, cognitive factors, organizational dynamics, and evaluation criteria after the adoption of Gen-AI tools. The analysis is based on Likert-scale responses collected.

Overall perceptions of productivity change after the adoption of Gen-AI were predominantly positive. Most responses were concentrated in the options indicating that productivity increased or increased significantly after adoption. Neutral perceptions, indicating no perceived change in productivity, were also reported, while no participants reported decreases in productivity. These results

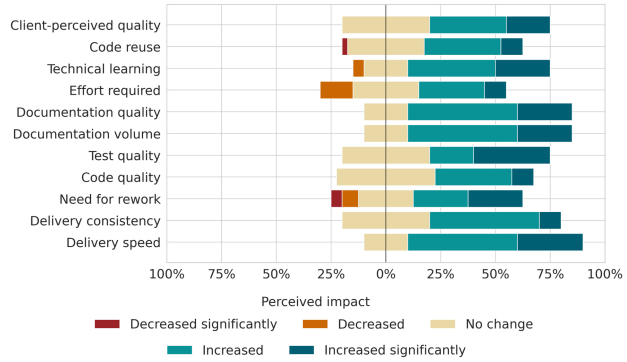

Figure 1: Perceived impacts of Gen-AI adoption on technical aspects of software project teams

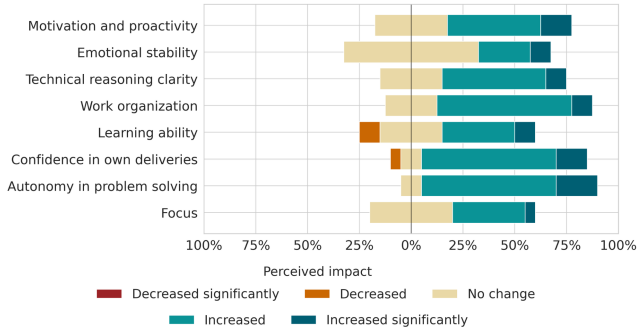

Figure 2: Perceived impacts of Gen-AI adoption on cognitive aspects of software project teams

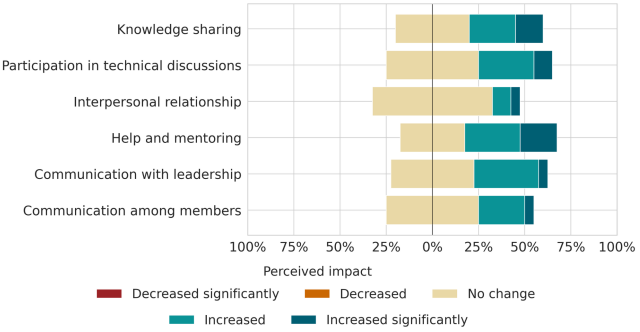

Figure 3: Perceived impacts of Gen-AI adoption on organizational aspects of software project team

suggest that participants generally associate Gen-AI adoption with positive impacts on software development productivity. However, the results also indicate that productivity gains are not interpreted only in terms of delivery acceleration, but increasingly in relation to how developers use, validate, and operationalize generated outputs.

Regarding technical aspects, Figure 1 presents participants' perceptions of technical factors impacted by Gen-AI adoption. Overall, delivery-related dimensions and software quality aspects showed predominantly positive distributions, particularly regarding delivery speed, delivery consistency, documentation support, and technical learning. In contrast, perceptions regarding rework and required effort presented greater variability, suggesting that although Gen-AI accelerates software production activities, part of the effort is redistributed toward validation, interpretation, and verification of generated outputs. These findings reinforce the emerging process of *productivity reinterpretation*, in which faster artifact generation does not necessarily eliminate cognitive and technical effort.

Figure 2 presents participants' perceptions regarding cognitive factors impacted by Gen-AI adoption. Most responses were concentrated between "no change" and "increased," particularly regarding autonomy in problem solving, confidence in deliveries, work organization, clarity of technical reasoning, and motivation and proactivity. These dimensions are associated with how developers interpret information, organize reasoning, evaluate alternatives, and make technical decisions during software development, rather than with the direct production of software artifacts. Accordingly, these results suggest that leaders perceive Gen-AI as influencing not only operational performance, but also how developers structure solutions, organize reasoning, and conduct technical activities. In contrast, focus, learning ability, and emotional stability presented greater variability, indicating that cognitive impacts may differ depending on individual experience levels and team contexts. Together, the predominance of perceived improvements in these cognitive dimensions suggests that productivity assessment becomes increasingly associated with cognitive mediation activities rather than only direct artifact production.

Organizational factors also presented distributions concentrated mainly between "no change" and "increased," as shown in Figure 3. Communication among team members, communication with leadership, mentoring, participation in technical discussions, and knowledge sharing showed predominantly positive perceptions after Gen-AI adoption. These findings suggest that collaborative activities increasingly involve discussing, validating, and refining generated outputs rather than only jointly producing software artifacts. Interpersonal relationships presented more neutral distributions and greater variability, indicating that social dynamics are less consistently affected by Gen-AI adoption. Overall, leaders perceive Gen-AI as influencing not only technical productivity, but also collaborative and interpretative dimensions of software development work.

The importance of developer attributes before and after the adoption of Gen-AI was also evaluated. Traditional competencies such as technical knowledge, domain knowledge, delivery speed, communication, collaboration, organization, and engagement remained consistently important according to participants. At the same time, attributes such as autonomy, critical thinking, and problem-solving ability showed indications of increased importance after Gen-AI adoption. These findings suggest that leaders increasingly value developers' ability to interpret, validate, and critically assess AI-generated outputs in addition to directly producing software artifacts. This shift reinforces the idea that productivity signals become progressively less centered on observable outputs alone and increasingly dependent on interpretative assessments of contribution and technical judgment.

Participants also evaluated the influence of Gen-AI on team deliveries. The results indicate that participants perceive Gen-AI as having a strong influence on team deliveries. The most participants selected the category "high" (61.9%), followed by "very high" (14.3%). Only one participant perceived a low influence, while no responses were registered for "very low." These findings suggest that leaders widely recognize Gen-AI as significantly affecting delivery dynamics, reinforcing perceptions of increased impact on development speed, execution support, and overall delivery performance in software engineering contexts.

Regarding productivity metrics, the results indicate that participants still rely on a combination of technical, delivery-oriented, and behavioral metrics to evaluate developer productivity. The most frequently adopted metrics include technical quality (95.2%), client-perceived quality (76.2%), delivery speed (71.4%), quantity of delivered tasks (66.7%), and collaborative behavior (66.7%). Regarding the perceived influence of Gen-AI on these evaluation metrics, most participants selected the positive range of the scale, particularly "positively" (52.4%) and "very positively" (14.3%). These findings suggest that leaders perceive Gen-AI as influencing not only operational performance indicators but also broader dimensions related to software quality, collaboration, and delivery effectiveness.

4.3 Qualitative Results

Gen-AI was frequently described as changing the meaning of productivity by accelerating execution while shifting effort toward validation and interpretation. P8 stated that Gen-AI contributes by "reducing the time for research and projects," associating productivity gains with faster information retrieval and solution drafting. Similarly, P3 explained that "the professional can filter exactly what they want to learn through Gen-AI without the need to search through documentation or forums," highlighting that productivity gains are tied to selective knowledge acquisition. These accounts suggest that productivity becomes linked to reduced exploration time and faster problem framing, rather than only increased delivery volume.

However, participants also described tensions between speed and quality. P14 reported that "the Gen-AI delivers very quickly [...] I have noticed lack of care when submitting pull requests, often missing developer testing and self-review," adding that developers sometimes "rewrite code almost like a ctrl + c / ctrl + v." This description connects acceleration with reduced review rigor and increased rework. In this sense, faster output does not necessarily translate into improved productivity, as validation effort becomes part of the work. Productivity therefore appears mediated by the balance between generation speed and verification effort, reinforcing the emerging process of *productivity reinterpretation* identified throughout the study.

The shift toward interpretation and validation also emerged in relation to technical judgment. P11 stated that "depending on the complexity of the work, it is possible to identify who is acting only at a basic level even using Gen-AI assistance," indicating that performance evaluation moves beyond output toward understanding. P15 similarly noted that "Gen-AI has patterns and I can differentiate individual authorship [...] without so much code optimization," suggesting that developers increasingly assess the structure and coherence of generated solutions. These accounts indicate that productivity becomes associated with the ability to interpret and refine Gen-AI-generated outputs rather than only directly producing software artifacts.

Changes in learning dynamics were also emphasized. P3 described Gen-AI as enabling targeted learning, stating that developers can "filter exactly what they want to learn," while P9 highlighted a contrasting perception, noting that "the human intellectual capacity is reduced to depend on Gen-AI." These responses illustrate a tension between cognitive support and dependency. Gen-AI reduces the effort required to access knowledge, but may also reduce the depth of engagement with technical problems. Productivity therefore becomes linked to how developers balance assistance and understanding.

Participants also described changes in collaboration practices. P4 explained that developers working in pairs now "debate the results of the problems solved by Gen-AI," indicating that interaction shifts from join production to evaluation and refinement of generated outputs. Collaboration becomes less centered on constructing solutions together and more focused on discussing alternatives, validating reasoning, and adjusting prompts. This suggests that Gen-AI alters the focus of collaboration without eliminating it.

Differences across experience levels also emerged. P14 described that a team composed of trainees experienced "lack of care [...] many times missing self-review [...] problems that end up blocking the pipeline," indicating that Gen-AI reliance may affect learning trajectories. This description suggests that productivity becomes uneven across seniority levels, as experienced developers may better interpret outputs while less experienced developers may rely more heavily on generated solutions.

Participants also indicated that evaluation practices are changing. P11 emphasized that it is possible to distinguish levels of understanding even with Gen-AI support, while P18 reported that expectations increased after Gen-AI adoption, stating that "No, I already got used to the increase." These responses indicate that productivity is no longer evaluated only by output, but also by how developers use Gen-AI and demonstrate understanding. Evaluation therefore becomes more interpretive and contextual, particularly in environments where observable artifacts no longer fully represent human contribution.

Taken together, the qualitative results indicate that Gen-AI reshapes productivity by shifting emphasis from production to mediation. Participants described productivity in terms of speed, validation effort, understanding, and interpretation of Gen-AI-generated outputs. These dimensions appear intertwined, suggesting that productivity in Gen-AI-assisted development becomes less directly observable and increasingly dependent on human judgment. Overall, the findings suggest an ongoing process of *productivity reinterpretation*, in which traditional output-based productivity signals become progressively less representative of actual developer contribution in Gen-AI-assisted software engineering environments.

5 Threats to Validity

As with any questionnaire-based study, this work is subject to threats to validity [18].

Regarding construct validity, one concern is whether the questionnaire adequately captures how Gen-AI adoption modifies productivity interpretation in software development teams. To mitigate this threat, the instrument was designed according to the research objectives, covering technical, cognitive, and organizational dimensions of productivity. It combined closed-ended, Likert-scale, and open-ended questions, and was refined through a pilot study with an experienced software team leader.

Internal validity may be affected by response bias, since the study relies on self-reported perceptions rather than direct observation of team performance. Participants may interpret productivity changes according to local practices, organizational culture, and personal beliefs regarding Gen-AI adoption. In addition, the study captures retrospective perceptions rather than directly comparing productivity evaluation before and after Gen-AI adoption over time. Social desirability bias may also influence responses involving merit attribution, dependency on Gen-AI, and fairness in evaluation. To mitigate these threats, participation was voluntary, responses were anonymous, and the questionnaire avoided leading formulations.

External validity concerns the generalization of findings. The study involved 20 leaders from software development and maintenance contexts in Brazil, providing exploratory evidence rather than statistical generalization. Participants were recruited through LinkedIn, email invitations, and personal contacts, which may have favored professionals more willing to participate in discussions about Gen-AI adoption. Consequently, leaders with less favorable perceptions of Gen-AI may be underrepresented in the sample. Organizational maturity, team structure, and intensity of Gen-AI adoption may also influence how productivity is interpreted. Therefore, the findings should be understood as recurring patterns perceived by practitioners rather than universal conclusions.

Conclusion validity concerns the interpretation of quantitative and qualitative evidence. The sample size limits the analysis to descriptive and interpretive perspectives, and qualitative coding may be influenced by researchers' interpretation. To mitigate this threat, the coding process followed a consistent structure aligned with the research questions, was reviewed by a second researcher, and combined quantitative and qualitative evidence to strengthen the consistency of the interpretations.

Despite these limitations, the study provides exploratory evidence of how software project leaders reinterpret productivity in Gen-AI-assisted software engineering environments and establishes a foundation for future studies involving broader samples and complementary empirical methods.

6 Conclusion and Future Work

This study investigated how software project leaders perceive developer productivity in Gen-AI-assisted software development environments. From the perspective of those responsible for evaluating developer performance, productivity extends beyond observable outputs such as code volume, commits, or delivery speed. Instead, project leaders increasingly associate developer productivity with activities related to validation, interpretation, technical judgment, and the effective use of Gen-AI tools.

The quantitative and qualitative results consistently indicate that leaders are also adopting new evaluation criteria, placing greater importance on autonomy, problem-solving ability, and the critical assessment of generated outputs while distinguishing developer contribution through interpretation and validation activities rather than output alone. Collectively, these results suggest that Gen-AI is reshaping expectations regarding developer performance, particularly in contexts where AI-generated outputs require continuous human supervision and validation.

This study contributes to ongoing discussions on software productivity by providing exploratory evidence of an emerging process of *productivity reinterpretation* in Gen-AI-assisted software engineering environments. Unlike previous studies that primarily examine productivity gains, developer perceptions, or operational efficiency associated with Gen-AI adoption, this work investigates how software project leaders interpret developer productivity and evaluate developer contribution in AI-assisted software development. Rather than proposing new productivity metrics, the study suggests that traditional output-based productivity signals alone may no longer fully capture developer contribution in contexts where software production increasingly depends on interaction, validation, and collaboration between developers and Gen-AI systems. Instead, Gen-AI appears to redistribute part of software work toward validation, interpretation, technical judgment, and other cognitive activities that are less directly observable through conventional software engineering metrics.

The findings also suggest that current transformations may represent an early stage of broader changes in software engineering work. While this study investigated environments where Gen-AI primarily acts as development assistance, emerging agentic software engineering scenarios may further intensify the separation between observable outputs and actual human contribution. In contexts involving autonomous agents, multi-agent collaboration, automated artifact generation, and AI-mediated development workflows, traditional productivity indicators may become even less representative of human reasoning, validation effort, and decision-making activities.

Future research should investigate how these emerging productivity signals evolve across different organizational contexts, levels of Gen-AI adoption, and software engineering practices. Longitudinal and observational studies may further clarify how evaluation criteria change over time and help establish evidence-based approaches for assessing developer contribution in increasingly AI-assisted software development environments.

Artifact Availability

Research artifacts are available at the following repository: Figshare.

Acknowledgements

We thank all the participants in the empirical study and the SPHERE Research Group members for their support. We would like to thank the financial support granted by CNPq - 313137/2025-0; CNPq 406506/2025-6; CAPES- Financing Code 001; Amazonas State Research Support Foundation - FAPEAM - through POSGRAD 2026-2027. We also acknowledge the use of Generative AI tools for text revision and language support. All content was reviewed and validated by the authors.

References

[1] Narallynne Araújo, Tiago Massoni, Camila Sarmento, Francielle Santos, and Ruan Oliveira. 2022. Investigating the relationship between software team leadership styles and turnover intention. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*. 106–111.

[2] Lenz Belzner, Thomas Gabor, and Martin Wirsing. 2023. Large language model assisted software engineering: prospects, challenges, and a case study. In *International conference on bridging the gap between AI and reality*. Springer, 355–374.

[3] Daniela S Cruzes and Tore Dyba. 2011. Recommended steps for thematic synthesis in software engineering. In *2011 international symposium on empirical software engineering and measurement*. IEEE, 275–284.

[4] Zixuan Feng, Sadia Afroz, and Anita Sarma. 2026. From gains to strains: Modeling developer burnout with genai adoption. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering*. 125–136.

[5] N Forsgren, MA Storey, C Maddila, T Zimmermann, B Houck, and J Butler. 2021. The SPACE of developer productivity: There's more to it than you think. acmqueue 19 (1), 20–48.

[6] Marcela Guerrero-Calvache and Giovanni Hernández. 2022. Team productivity in agile software development: a systematic mapping study. In *International Conference on Applied Informatics*. Springer, 455–471.

[7] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems* 43, 2 (2025), 1–55.

[8] Barbara A Kitchenham and Shari L Pfleeger. 2008. Personal opinion surveys. In *Guide to advanced empirical software engineering*. Springer, 63–92.

[9] Moez Krichen and Mohamed S Abdalzaher. 2025. Generative AI for Code Development: Advancements, Limitations, and Prospects. In *Generative AI and Large Language Models: Opportunities, Challenges, and Applications: Volume 1*. Springer, 217–249.

[10] Mahei Manhai Li, Ernestine Dickhaut, Olivia Bruhin, Hendrik Wache, and Pauline Weritz. 2024. More than just efficiency: impact of generative AI on developer productivity. In *30th Americas Conference on Information Systems, AMCIS 2024*. Association for Information Systems.

[11] Amr Mohamed, Maram Assi, and Mariam Guizani. 2025. The impact of LLM-assistants on software developer productivity: A systematic literature review. *arXiv preprint arXiv:2507.03156* (2025).

[12] Francisco Victor da S Pinheiro, Emanuel F Coutinho, Marcelo M da Silva, Sidartha A Lobo de Carvalho, and Rossana MC Andrade. 2025. Investigando a Integração Estrutural de LLMs em Ecossistemas de Software: Um Estudo com Modelagem SSN. In *Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, 832–838.

[13] Sergio Rico and Lena-Maria Öberg. 2025. Challenges and Opportunities for Generative AI in Software Engineering: A Managerial View. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*. 1338–1344.

[14] Sabrina Rocha, Rodrigo Feitosa, Larissa Galeno, and Guilherme H Travassos. 2025. A Metaprotocol For a Family of Rapid Multivocal Reviews of Generative AI in the Software Industry. In *Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, 804–810.

[15] Reine Santos, Igor Steinmacher, Tayana Conte, Ana Carolina Oran, and Bruno Gadelha. 2025. AI-Generated User Stories: Are They Good Enough?. In *Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, 741–747.

[16] Rasmus Ulfsnes, Nils Brede Moe, Viktoria Stray, and Marianne Skarpen. 2024. Transforming software development with generative ai: Empirical insights on collaboration and workflow. In *Generative AI for effective software development*. Springer, 219–234.

[17] Andreas Vogelsang. 2024. From specifications to prompts: On the future of generative large language models in requirements engineering. *IEEE Software* 41, 5 (2024), 9–13.

[18] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-29044-2