

SPECIAL ISSUE ARTICLE

SMS | Strategic Management Journal

WILEY

Strategy and Artificial Intelligence Special Issue

The impact of generative artificial intelligence on innovation: Evidence from software products

Erdem Doğukan Yılmaz¹  | Tim Meyer² ¹Department of Family Business and Entrepreneurship, INSEAD, Fontainebleau, France²Institute of Management & Strategy, Center for Financial Services Innovation, University of St. Gallen, St. Gallen, Switzerland**Correspondence**

Erdem Doğukan Yılmaz, Department of Family Business and Entrepreneurship, INSEAD, Fontainebleau, France.

Email: erdemdogukan.yilmaz@insead.edu

Abstract

Research Summary: We study the impact of generative artificial intelligence (GAI) tools on product-level innovation outcomes in the context of software products. Specifically, we illustrate how GAI can alter the direction of innovation by shifting the activities of developers away from generational innovation and toward original innovation, which may be new to the market but not necessarily more novel than previous innovations. We argue that this shift is driven by GAI's ability to facilitate tasks in both the ideation and implementation of software products, which enables some developers to create software products that were previously beyond their reach, while allowing others to reallocate their effort with respect to different activities. Our analyses in the context of browser add-ons provide empirical evidence for these arguments. We discuss implications for innovation research and the generalizability of our findings to other domains.

Managerial Summary: Generative AI tools are changing how organizations innovate. Studying the

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2026 The Author(s). *Strategic Management Journal* published by John Wiley & Sons Ltd.

Firefox browser add-on ecosystem, we find that after ChatGPT's release, the number of new add-ons increased by 34%, driven by both existing and first-time developers. However, developers simultaneously reduced improvements to their existing add-ons by 20%, redirecting effort toward new projects. While new add-ons drew on a broader set of knowledge domains, they were not substantially different from what already existed. This suggests that generative AI helps developers efficiently combine existing knowledge rather than produce truly novel ideas. For platform managers and business leaders, these findings highlight that although Generative AI can democratize product creation and accelerate new product launches, it may require new strategies to maintain product quality and encourage genuine novelty.

KEYWORDS

digitization, generative artificial intelligence, innovation

1 | INTRODUCTION

Generative artificial intelligence (GAI) and particularly large language models (LLMs) are reshaping the economics of task performance in the innovation process. Specifically, research has shown that these tools can facilitate specific tasks by substituting or complementing human efforts (Acemoglu & Restrepo, 2024; Brynjolfsson et al., 2025; Noy & Zhang, 2023; Yilmaz, Naumovska, & Aggarwal, 2023). However, by facilitating specific tasks, these tools may also enable individuals to engage in new types of tasks. They may enable inexperienced individuals to begin developing products that they were unable to develop before, while allowing experienced individuals to shift their effort with respect to different innovation tasks. Yet, the effect of these shifts on actual innovation outcomes and the broader trajectory of innovation remains unclear. In this paper, we therefore draw on the context of software product development, a setting where the impact of GAI is likely most immediate, to study how the introduction of GAI impacts the types of products that individuals choose to develop and thus the direction of innovation, as well as how this is driven by experienced and inexperienced individuals.

Recent advancements in artificial intelligence (AI), particularly the emergence of GAI tools like *ChatGPT*, have sparked intense interest in how these technologies might reshape innovation across various domains (Agrawal et al., 2022; Cockburn et al., 2018). Similar to other general-purpose technologies, such as the steam engine or computers, GAI is characterized by its wide applicability and capacity to spur complementary innovations. However, GAI differs from other forms of general-purpose technologies in two main ways. First, GAI can assist individuals also in non-routine cognitive tasks, such as creating or refining ideas (He et al., 2024), which have often been seen as the sole province of human intelligence (Argote et al., 2021;



Brynjolfsson et al., 2018; Yilmaz, Naumovska, & Aggarwal, 2023). For this reason, deep learning models—the foundation of GAI—have been described as an “invention that makes inventions” (IMIs), that is, a technology that does not merely improve productive efficiency but fundamentally reshapes the innovation process by facilitating the creation of innovations itself (Agrawal, Gans, & Goldfarb, 2019; Cockburn et al., 2018). Second, in many settings, the generative nature of GAI can also assist individuals in the implementation of ideas into actual products. This is particularly true for settings such as computer software, apps or online content where the innovation outcome is digital, and relatively complex output can be generated from relatively simple prompts. Taken together, these attributes position GAI not merely as another general-purpose technology (GPT) for efficiency gains, but rather as a technology with the potential to change what kinds of innovations are pursued and who is able to pursue them.

Existing studies on GAI have studied its effects on creativity and productivity (Boussioux et al., 2024; Brynjolfsson et al., 2025; Doshi & Hauser, 2024; Girotra et al., 2023; Noy & Zhang, 2023), highlighting that GAI can outperform (and thus potentially substitute) humans in certain tasks of the innovation process, while substantially increasing (and thus complementing) their capabilities or their productivity in other tasks. Yet, we still know comparatively little about how these task-level effects translate into changes in the portfolio of products that innovators choose to build and maintain once GAI becomes widely available, leaving open the question of how it may alter actual product-level outcomes at scale. As GAI becomes available to individuals in real-world settings, these individuals will likely reallocate their innovation efforts with respect to different tasks and ultimately different products they develop. GAI may therefore not merely lead to a shift in the level of innovation or productivity but also change the trajectory of innovation.

We study the impact of GAI tools on actual product-level innovation outcomes in the context of software products, which is one of the largest and most innovative sectors in the economy, and one that has therefore been studied extensively by scholars in the management field (Bennett & Hall, 2020; Chatterjee, 2017; Ethiraj et al., 2005; Tiwana, 2008; Zahavi & Lavie, 2013). While the effect of GAI can manifest in various innovation contexts, its imprint is likely most immediate in digital innovation, given the higher level of knowledge generativity and the close alignment between GAI's digital output and the digital nature of products. We argue that by facilitating certain tasks, GAI assistance¹ enables some software developers to develop products previously beyond their reach, while allowing others to reallocate their efforts across different activities. Specifically, we begin by arguing that the availability of GAI tools leads to an increase in the number of *original*² software products being introduced due to its ability to facilitate idea generation and implementation. We posit that this increase in the rate of original products being introduced stems partly from GAI enabling new developers to generate and implement ideas, despite their lack of domain knowledge, experience, and resources (Conti et al., 2014; Jeppesen & Lakhani, 2010).

¹We borrow the term “GAI assistance” from previous research (Jia et al., 2024), which has used it to refer to individuals being assisted by artificial intelligence tools in the innovation process.

²We use the term *original* (software) products to refer to products that are newly introduced to the market. We distinguish these original products from *generational* innovation (i.e., improvements to existing products), which consist of refinements to previously introduced products. Similar categorizations have been used by previous research (e.g., Ahuja et al., 2013) (see Section 3.3). This terminology is related to, yet distinct from, categorizations such as radical/incremental innovation (Dewar & Dutton, 1986) or pioneering/incremental innovation (Ali et al., 1993), which are often understood as differing primarily in their degree of novelty. Note that not all *original* products may be equally novel.

Perhaps more importantly, we also argue that GAI may affect the *direction* of innovation activity in two ways: by affecting how developers allocate their (limited) resources to original (i.e., newly launched) products or to improvements of existing products (i.e., *generational* innovation), and by affecting the scope of knowledge that original products draw on. First, we propose that GAI assistance may shift product creation efforts by existing developers away from product improvements (i.e., *generational* innovation) and toward the development of original products. This shift, we argue, occurs because GAI frees up resources by reducing task-specific costs, while at the same time facilitating idea generation, which allows developers to pursue the development of original products previously constrained by resources (Greve, 2003; Nohria & Gulati, 1996). Additionally, established developers may have an incentive to focus on original products rather than mere improvements of existing ones in order to fight competition from new developers who start creating original products as well (Ahuja et al., 2013).

Second, we argue that GAI assistance will increase the scope of knowledge that original products draw on by allowing individuals to combine knowledge from a larger number of different domains. This is consistent with the idea that GAI can draw on higher-dimensional search spaces and explore more diverse territories within these search spaces (He et al., 2024). At the same time, these products may not necessarily be more novel (i.e., they may not be substantially different from existing products that were introduced before [Dahlin & Behrens, 2005; Katila & Ahuja, 2002; Rosenkopf & Nerkar, 2001]). We argue that this is because GAI tends to surface high-probability combinations that are well-represented in prior data, and individuals may struggle to correctly assess their true novelty. Hence, while originality and novelty are often assumed to correlate, we propose that GAI may differentially affect them by enabling the creation of original products without necessarily increasing their novelty. We think of this phenomenon as GAI enabling “efficient recombination”, that is, GAI allowing individuals to access and combine knowledge from diverse domains with lower cognitive and resource costs, without necessarily shifting innovators into completely uncharted knowledge territory (Fleming, 2001; Gruber et al., 2013).

We test our theoretical reasoning using data on the *Firefox* browser add-on platform developed by *Mozilla Corporation*. This platform allows third-party developers to develop add-ons (sometimes referred to as extensions) to the *Firefox* internet browser and has been used as a research setting by prior work in the management literature (e.g., MacCormack et al., 2006; Song et al., 2018; Tiwana, 2015; van der Geest & van Angeren, 2023). These add-ons are software products that enhance user experience or solve specific problems that users face. While add-ons may seem incremental, they often represent novel combinations of existing technologies or the application of new ideas to improve browser functionality. This setting is particularly useful for our purpose as it lets us clearly trace which types of new products are being developed.

We employ an event study centered around the introduction of *ChatGPT* to study the effect of GAI assistance on software product introductions. Using a number of different econometric approaches that draw on different sets of control groups, we first study the effect of GAI on the number of original products (i.e., new add-ons) introduced and find empirical evidence for a positive effect. The results from panel differences (PD) analyses reveal a 34.4% increase in the introduction of new browser add-ons, originating from both new and existing developers. Consistent with our reasoning, after the introduction of *ChatGPT*, add-ons typically span more categories but do not differ in their similarity to previous products, compared to the control group. This indicates that while the quantity of new products has increased and developers may be more likely to combine knowledge from different domains, the novelty of the products does not



seem to have increased. Consistent with the idea that GAI may shift the focus from improving existing products to creating original ones, our analyses also reveal that, following the introduction of *ChatGPT*, the rate of updates of existing add-ons declined by 20.3%. We then go on to conduct a rich set of additional in-depth analyses that help us further unpack the effect of GAI on new products and their developers. We also rule out alternative explanations, such as demand-side (rather than supply-side) factors or changes in the *Mozilla* ecosystem driving our findings. To better understand the underlying mechanism and provide support for our theoretical reasoning, we complement our main dataset with qualitative developer interviews and quantitative data from an exploratory survey of 33 *Firefox* developers. The survey focuses on developers' experiences with and perceptions of GAI, including its effect on their ideation process, implementation workflow, development timelines, and competitive environment. The results from the survey are consistent with the theoretical mechanisms we propose. We also discuss how our findings may generalize to other settings.

This paper makes two main contributions. First, while previous literature has primarily focused on how GAI can complement or even substitute human individuals in certain tasks (Boussiou et al., 2024; Doshi & Hauser, 2024; Girotra et al., 2023), we study how, as a reaction to such complementarity or substitution effects, individuals shift their innovation effort with respect to different innovation tasks. Specifically, our work shows how GAI affects the type and direction of innovation that individuals pursue, and thus goes beyond a mere study of the efficiency of innovation efforts. Where previous research has examined how GAI affects performance on isolated tasks, our study shows how these task-level effects cascade into broader reallocation decisions that transform both what is developed (more original products) and what is neglected (fewer improvements to existing products). We also illustrate how the idea of “efficient recombination” allows us to understand why there may be a higher number of original products that are not necessarily more novel. Unlike previous studies, we focus on GAI's impact on real-world innovation outcomes, thus extending previous research that has often focused on *individual elements* of the innovation process, such as problem-solving or creativity (e.g., Boussiou et al., 2024; Doshi & Hauser, 2024). Our data on real-world software products enables us to observe which types of products are developed and to track changes in the trajectory of innovation. In doing so, we also address calls for more field-based studies of AI's impact on organizational outcomes (Raisch & Krakowski, 2021; Von Krogh, 2018).

Second, we contribute to the literature on how new technologies affect innovation trajectories and, more specifically, to existing research on general-purpose technologies and innovation (e.g., Bresnahan & Trajtenberg, 1995; Helpman, 1998). While previous research on general-purpose technologies has highlighted that their adoption and evolution are often slowed down by coordination problems with the providers of application products (Bresnahan & Trajtenberg, 1995), we illustrate how GAI availability can have an immediate effect on innovation and, more specifically, on the direction of innovation. We attribute this to the fact that deep learning, and GAI in particular, is not just a GPT (Eloundou et al., 2024) but an invention of a method of inventing (Cockburn et al., 2018), which “enable[s] a new approach to innovation itself, by altering the ‘playbook’ for innovation in the domains where [it] is applied” (Cockburn et al., 2018, p. 120) and directly lowers the cost of both ideation and implementation (Cockburn et al., 2018; Eloundou et al., 2024). This also speaks to a growing body of research that has begun to study how technologies related to automation (Furman & Teodoridis, 2020) or data-driven predictions (Tranchoero, 2024) affect which types of innovations may be pursued. We complement and extend this work by showing how GAI's unique characteristics can shift the focus toward more original product development, as opposed to improvements of existing

products. Our results also contribute to the ongoing debate about whether AI will primarily support exploitation of existing knowledge or enable more exploratory innovation (Agrawal, Gans, & Goldfarb, 2019; Furman & Teodoridis, 2020) by showing that, while developers focus more on creating original products, these products are not necessarily more novel.

2 | BACKGROUND LITERATURE

2.1 | Innovation

Innovation is typically understood as a process comprising two main phases which, depending on the setting, may be more or less intertwined: Ideation, where innovators search for ideas or opportunities, and implementation, where they implement these ideas into products (Kijkuit & van den Ende, 2007; Schumpeter, 1934).

In the first phase, innovators typically aim to generate new knowledge that goes beyond their existing knowledge base in order to create inventions that can be transformed into products in the second phase. This can be achieved, for instance by drawing on multiple elements of existing knowledge and potentially recombining it in new ways (Amabile, 1996; Arts & Veugelers, 2015; Fleming, 2001; Schumpeter, 1934) or by adding new knowledge to existing knowledge (Nagle & Teodoridis, 2020). In the context of software product development, the first phase of the innovation process typically involves developers generating ideas for products, which may, for instance, contain new features or recombine features from existing software products.

The second phase of the innovation process involves the implementation of the inventions from the first phase into products. Typically, this phase of the process requires additional resources, such as financial resources, or specific knowledge related to the implementation process itself (Chandy et al., 2006). In some settings, this knowledge can be fairly distinct from the knowledge required in the first phase (Nerkar & Roberts, 2004). In software product development, this phase mainly involves developers implementing their ideas into actual computer code.

The overall direction (or trajectory) of innovation in a given setting (e.g., software development) is shaped by the types of products that are developed by innovators (Dosi, 1982). For instance, innovators may have to decide between allocating their efforts to original innovations (e.g., a new product) and allocating them to improvements of existing products (Ahuja et al., 2013; Chen, Zhang, et al., 2022). Typically, the direction of innovation that innovators pursue is shaped by their knowledge (Chatterji & Fabrizio, 2014) or assets (Wu et al., 2014). However, due to resource constraints, pursuing multiple directions with the same intensity is typically not an option.

Moreover, while products may be original, that is, they may not have been available on the market before, they may vary in their novelty, that is, the degree to which a product diverges from existing products (Dahlin & Behrens, 2005; Katila & Ahuja, 2002; Rosenkopf & Nerkar, 2001). An innovator may launch a product that is original (in the sense that it has not previously existed on the market) but not necessarily novel (as it may be very similar to other existing products on the market).³ This may be the case, for instance, if the new product mainly

³Previous literature has often described the novelty of innovations using the “radical vs. incremental” distinction, which is sometimes thought of as reflecting a theoretical continuum of novelty (Dewar & Dutton, 1986).



(re)combines knowledge from multiple existing knowledge domains without making the resulting product significantly different from existing products and without, in turn, shifting innovators into completely uncharted knowledge domains.

2.2 | Generative AI

The availability of AI can significantly alter the way individuals complete their tasks in the innovation process. The broader literature on AI and innovation, while emerging, points toward a consensus that AI represents a GPT, characterized by its wide-ranging applications across diverse industries and its capacity to significantly enhance innovation (Agrawal, McHale, & Oettl, 2019; Cockburn et al., 2018; Eloundou et al., 2024; Goldfarb et al., 2023). General-purpose technologies, such as the steam engine and electricity before it, have fundamentally transformed economies by enabling new forms of production, processes, and services. AI, however, distinguishes itself by automating cognitive tasks, once believed to be the sole province of human intelligence (Brynjolfsson et al., 2018; Yilmaz, Naumovska, & Aggarwal, 2023).

An important point in this regard lies in the distinction between GAI, which has seen broad adoption only very recently, and more traditional types of AI, such as *predictive* AI (PAI). Traditionally, AI systems were often designed to use patterns in historical data to forecast future outcomes or classify future events. Through their ability to process great amounts of data, they can surpass human information processing constraints, which has enabled a myriad of applications that support humans in uncovering new areas for investigation (Cockburn et al., 2018; Crafts, 2021) or in predicting which combinations of knowledge will be most pertinent for a given task (Agrawal, McHale, & Oettl, 2019). For instance, AI applications exist in pharmaceutical R&D (Mamoshina et al., 2016), and their adoption was found to influence a firm's exploration of new drug-target pairs for preclinical studies in certain cases (Lou & Wu, 2021). However, despite being remarkably helpful in prediction and discovery tasks, these AI systems have been considered inferior to humans in terms of their ability to handle creative tasks (Argote et al., 2021).

This contrasts with GAI, exemplified by tools such as *ChatGPT*, which leverages large amounts of data to generate content that mimics human-like understanding and creativity. The advantage of generative GAI in enhancing creativity (Jia et al., 2024) stems from its ability to not only process and analyze data at a scale beyond human capability but also to combine information across different domains and thus generate ideas and concepts that were previously unthought of (He et al., 2024). Moreover, GAI allows users to transform such ideas into tangible outcomes, including text, images or, as in our setting, computer code, which can be generated with limited prior experience through relatively simple prompts. The examples of GAI for text are LLMs. These models identify statistical interrelations between words and learn to predict subsequent words in a sentence based on preceding context, thus generating text that is both grammatically coherent and semantically rich. LLM's use of transformer architecture and an ample quantity of unlabeled data for pre-training enable them to understand the nuances of human language and computer code without explicit supervision.

These unique capabilities position GAI not merely as a GPT but as an “invention that makes inventions” (IMI) (Cockburn et al., 2018; Griliches, 1957), that is, a technology that does not only increase the efficiency of production or innovation, but “enable[s] a new approach to innovation itself, by altering the ‘playbook’ for innovation in the domains where [it] is applied” (Cockburn et al., 2018, p. 120). In other words, while previous general-purpose technologies

such as the steam engine may have led to innovation in the focal technology (i.e., the steam engine) and its application sectors (e.g., manufacturing machinery powered by steam engines), in the context of GAI, the technology (i.e., GAI) itself creates innovations by supporting the ideation and implementation phases of innovation. As such, GAI is remarkably different from other general-purpose technologies.

While previous research suggests that general-purpose technologies can reshape innovation by changing its direction through shifts in relative costs and returns across innovation activities (Bresnahan & Trajtenberg, 1995; Brynjolfsson et al., 2021), it has also highlighted that the effect of general-purpose technologies is often delayed. This can be attributed to coordination problems between the GPT and its application sectors, where a lack of incentives in one of these sectors can result in system-wide underinvestment (Cockburn et al., 2018). More generally, scholars have observed that general-purpose technologies often lead to little productivity gains at the beginning, as they often require large investments (Brynjolfsson et al., 2021) to develop their full potential. Sometimes, this phenomenon has been referred to as the “productivity J-curve” (Brynjolfsson et al., 2021). Given GAI’s distinctive nature as an IMI, which itself enables innovation, its dependency on complementary innovation is arguably lower. Therefore, it remains unclear how patterns of other general-purpose technologies translate to GAI and whether one can observe a (perhaps more immediate) effect on the type and direction of innovation that is conducted.

A growing stream of research has started to study how the availability of new technologies that assist the innovation process can significantly affect the rate and direction of innovation. For instance, it has been shown that the availability of technologies that automate certain research tasks can increase overall idea generation (Furman & Teodoridis, 2020) and that data analytics is more valuable in process innovation than in the development of entirely new technologies (Wu et al., 2020). Studies on PAI and data-driven predictions more generally have shown that these technologies may increase innovation success under certain conditions (Agrawal et al., 2024), particularly if they are complemented by domain knowledge of the innovator (Trancharo, 2024). In the context of GAI and innovation, a number of studies have used laboratory experiments (Doshi & Hauser, 2024; Noy & Zhang, 2023) or field experiments (Boussioux et al., 2024) to study how GAI may complement or substitute specific tasks that are typically performed by humans. Importantly however, these studies typically focus on *individual elements* of the innovation process. For instance, Doshi and Hauser (2024) focus on creative output, Boussioux et al. (2024) study problem-solving, and Trancharo (2024) focuses on the identification of innovation opportunities. What remains less understood is how, as a reaction to complementarity or substitution effects, individuals reallocate their innovation efforts and how task-level effects ultimately cascade into changes in the actual products that are developed. Our data on real-world innovation outcomes enables us to extend prior research by studying how individuals’ reallocation of innovation efforts with respect to different tasks is reflected in the types of innovations that are introduced to the market and, ultimately, the trajectory of innovation.

3 | THEORY AND HYPOTHESES

In this section, we develop predictions about the effect of GAI assistance on innovation in the context of software products. Software is a sector that has gained major relevance in most economies over the past decades and has been studied by numerous studies in the field of

management (Bennett & Hall, 2020; Chatterjee, 2017; Ethiraj et al., 2005; Tiwana, 2008; Zahavi & Lavie, 2013). We discuss the generalizability of our findings to other sectors in Section 6.3.

3.1 | Generative AI and the rate of original software product introductions

GAI has the potential to assist developers in both phases of the innovation process, that is, in the ideation and implementation phases, in multiple ways.

In the first phase, ideation, individuals usually face two barriers when they aim to develop new ideas. First, they must overcome information processing constraints (Levinthal & March, 1993) that limit the amount of information on new opportunities the individual may pursue. These constraints stem from individuals' biologically limited mental capacities to absorb or process information. The second barrier results from ineffective or localized search routines (Levinthal & March, 1993). This barrier suggests that individuals typically search for solutions in knowledge domains related to their own existing knowledge (Weisberg, 1999), implying that most solutions will likely be incremental in their innovative thrust (Csaszar & Levinthal, 2016) and that individuals may be limited in their ability to draw on other knowledge domains. GAI may help developers mitigate information processing constraints and engage in exploratory searches beyond their personal expertise. This is because GAI can draw on higher-dimensional search spaces (He et al., 2024) which facilitates access to new knowledge domains that individuals may otherwise not have been aware of. This can help developers generate ideas that they can then transform into actual products in the implementation phase. Indeed, studies indicate that GAI may help in generating ideas that are more valuable (Boussieux et al., 2024) and exhibit higher quality (Girotra et al., 2023) than human ideas. Similarly, with GAI assistance, developers may find it easier to experiment with multiple ideas and then select the most promising one out of a given pool.

In the second phase of innovation, that is, implementation, GAI can support developers as well. Specifically, GAI can assist developers by providing advice and instructions on how to perform certain tasks that may be necessary for the implementation of ideas, or by directly implementing ideas into products (Noy & Zhang, 2023), for example, software code. More generally, there is evidence that GAI assistance can significantly increase the productivity of humans (Brynjolfsson et al., 2025; Noy & Zhang, 2023). For example, GAI tools can rapidly pinpoint software bugs or create initial software code drafts, thus allowing developers to focus on editing (Noy & Zhang, 2023). This can increase the number of ideas that are implemented in a given timeframe and may reduce the (financial) resources that are required for implementation. More generally, developers might be able to rely on GAI for some of the tasks that would have otherwise been performed by themselves or by their collaborators.

As GAI assistance can positively affect both the ideation and implementation phase of software product development in multiple ways, we expect that the rate of original software products being introduced will increase:

Hypothesis 1. The availability of GAI assistance is associated with an increase in the number of original software products.

3.2 | Generative AI, entry barriers, and the source of innovation

Previous research has distinguished between two broad patterns of innovation: a deepening pattern where innovation is mostly carried out by established innovators, and a widening pattern where entry barriers are relatively low, and the innovation base is continuously expanded through the entry of new innovators (Malerba & Orsenigo, 1996). The introduction of GAI assistance may significantly impact both patterns, as the increase in the rate of original products introduced after the availability of GAI assistance may not only be driven by existing developers, but also by newly entering developers.⁴

In the first phase of innovation, newly entering developers typically lack knowledge and experience in the given domain compared to existing developers (Shane, 2000), which may limit their ability to generate original products. However, GAI assistance may help new developers to overcome this lack of domain-specific knowledge. Indeed, there is evidence that GAI assistance can significantly increase output quality for actors with limited ability (Noy & Zhang, 2023), which is likely the case for new developers that have not accumulated sufficient knowledge. Importantly, studies have found that GAI assistance is not merely a substitute for lack of knowledge, but helps inexperienced individuals move along the experience curve more quickly (Brynjolfsson et al., 2025).

In the second phase of the innovation process, there are important benefits for newly entering developers as well. First, much like new developers lack domain-related knowledge in the first phase, they may also lack knowledge on how to implement their ideas in the second phase, that is, knowledge on how to write effective software code. Consider an individual who aims to introduce a personalized nutrition app but has limited experience. GAI assistance may not just help this individual by providing her with domain-related knowledge (e.g., what the typical features of nutrition apps are) and how to (re)combine this knowledge in creative ways, but also in implementing the resulting ideas by transforming plain language objectives and requirements into actual computer code. Similarly, GAI may empower developers with creative ideas but limited resources, by allowing them to be more efficient with their tasks and thus reducing the amount of required resources. This reasoning is supported by findings that inexperienced individuals can not only achieve better results with the help of GAI, but also complete their tasks faster (Noy & Zhang, 2023). Therefore, we expect the increase in the rate of original product introduction to be driven not only by the intensive margin (i.e., *existing* developers introducing a higher number of original products) but also by the extensive margin (i.e., more *new developers* starting to develop original products). Hence, we argue that:

Hypothesis 2. The availability of GAI assistance is associated with an increase in the number of original software products introduced by new developers.

3.3 | Generative AI and the direction of innovative activities in software products

Beyond increasing the rate of software product introductions, GAI may also affect the direction of innovation activity. In particular, we theorize about how it may do so in two ways: by affecting how developers allocate their (limited) resources to original products or to improvements of existing products, and by affecting the scope of knowledge that original products draw on.

⁴In Appendix D.8, we study potential differences in GAI adoption between new and existing developers (Allen & Choudhury, 2022) and find no evidence for differences between the two groups.



We refer to the development of completely new products (i.e., products that have not existed on the market before) as “original innovation”, while improvements to existing products (e.g., software *updates*) are often referred to as “generational innovation” (Ahuja et al., 2013; Chen, Zhang, et al., 2022). Typically, innovators cannot pursue both directions with the same intensity, as their resources and their attention are limited (Ocasio, 1997). However, the introduction of new technologies can significantly alter what developers focus on.

While seemingly less important than entirely new products, generational innovations are valuable for innovators, as they can provide opportunities for knowledge expansion (Helfat & Raubitschek, 2000), grow the market share (Banbury & Mitchell, 1995), and ultimately ensure better performance compared to rivals (Lawless & Anderson, 1996). They play a particularly important role in the context of software products, where an original product is often followed by a series of updated versions of this product (Chen, Zhang, et al., 2022; Miric et al., 2019; Turner et al., 2013). In a digital context, introducing improvements on a regular basis can serve as an important means of protecting one's original products against competitors, as other forms of intellectual property protection are less effective in these settings (Miric et al., 2019; Yilmaz, Naumovska, & Miric, 2023).

The availability of GAI assistance may shift the balance between original products and improvements being introduced. In general, GAI's ability to broaden the scope of knowledge and combine knowledge from different domains is arguably more beneficial for original products, as they are often associated with moving away from developers' existing knowledge and entering new innovation trajectories (Ahuja et al., 2013). Conversely, generational innovation is typically seen as the result of focusing on a relatively narrow innovation trajectory (Ahuja et al., 2013), implying that the potential benefit from GAI assistance is lower.

Moreover, as we discussed above, GAI significantly reduces the cost associated with idea generation and implementation, allowing developers to explore ideas they may not have considered before and to do so with lower resource input. Together, this not only enables developers to pursue a wider range of (original) projects, including those that were previously deemed too resource-intensive or risky, but also frees up slack resources, which have been shown to lead to more openness toward experimentation (Greve, 2003; Nohria & Gulati, 1996).

Besides these direct effects of GAI assistance, there may also be a more indirect effect driven by the increasing rate of original products being introduced, particularly by new developers. As developers observe this increase in competition, they may increasingly turn to more original innovation in an effort to avoid being left behind. In the absence of competition, developers may have little incentive to introduce new products that may even cannibalize their existing products (Chandy & Tellis, 1998) and may resort mainly to improvements to existing products. In the face of increasing competition however, developers may change the direction of their innovation efforts (Wen & Zhu, 2019) and prioritize original product development over mere improvements, hoping to escape competition and to differentiate themselves from others (Miric & Jeppesen, 2020). Therefore, we expect that the increase in original products comes at the expense of a decrease in improvements to existing products.

Hypothesis 3. The availability of GAI assistance is associated with a decrease in the number of improvements to existing software products.

Research has long emphasized that innovation often emerges from recombining different knowledge elements (Fleming, 2001; Schumpeter, 1934). However, recombination typically involves a search process through different knowledge domains to identify knowledge elements that can be recombined (Fleming & Sorenson, 2004). Typically, this search process is costly, as

individuals need to invest substantial time, effort and resources to access new knowledge domains and familiarize themselves with them (Teodoridis et al., 2019; Xiao et al., 2022). This, in turn, creates important frictions in the innovation process (Fleming, 2001).

GAI can draw on higher-dimensional search spaces and explore more diverse territories within these search spaces than human individuals (He et al., 2024). For individuals who are assisted by GAI, this reduces the cost of *accessing* new knowledge domains and allows them to include a broader set of knowledge elements in their innovation, which they may otherwise not have been aware of. Developers may leverage GAI to combine knowledge elements from different domains more quickly or at a larger volume, yielding *original* (i.e., newly launched) products at scale. We refer to this process as “efficient recombination”. At the same time, while GAI helps individuals draw on and combine a broader set of knowledge elements, the resulting combination may not necessarily be *novel* (i.e., differ significantly from previous products or solutions [Katila & Ahuja, 2002; Rosenkopf & Nerkar, 2001]). We argue that this is because the burden shifts toward identifying truly novel combinations, which can be challenging for individuals.

GAI models are trained on existing corpora of knowledge and tend to surface high-probability combinations that are well-represented in prior data, rather than combinations where observational data is scarce (Zahavy, 2026).⁵ Hence, while individuals can readily access and combine a broader set of knowledge domains, these combinations will often be relatively close in structure to existing ones. This resembles the “streetlight effect”, where the attention of individuals is more heavily drawn to more accessible rather than more valuable areas (Hoelzemann et al., 2024; Kim, 2025). Moreover, as individuals enter broader sets of knowledge domains, they are increasingly confronted with knowledge they are less familiar with and on which they may lack in-depth expertise (Keijl et al., 2016). This, in turn, can reduce their ability to identify truly novel combinations (Fleming, 2001; Kaplan & Vakili, 2015) and they may either evaluate the novelty of potential options incorrectly or opt for lower-risk adaptations of relatively familiar elements (Argote et al., 2021). This tendency imposes an additional selection layer on the set of combinations suggested by GAI.

Taken together, we expect developers to draw on a broader set of knowledge domains when creating original products with GAI assistance, even if the fundamental novelty of these products—as measured by their distance from existing solutions—may not increase proportionally. Consequently, we expect that:

Hypothesis 4. The availability of GAI assistance is associated with a broader scope of knowledge domains that original software products draw on.

4 | METHODS

4.1 | Setting

To test our hypotheses, we use data on the *Mozilla Firefox* (or simply *Firefox*) browser add-on platform, an open-source internet browser platform developed by the *Mozilla Corporation*, a

⁵Naturally, GAI models improve over time in their ability to explore and combine different areas of knowledge, and individuals may develop more effective strategies for interacting with these models. The limitations of GAI models may thus be partly contingent on the current state of technology, and the proposed mechanism may be more pronounced in the earlier generations of GAI tools.



global community of technologists, thinkers, and builders working collaboratively. We thus follow previous work in the management literature, which has used this setting before (e.g., MacCormack et al., 2006; Song et al., 2018; Tiwana, 2015; van der Geest & van Angeren, 2023). Browser add-ons serve as small software products that are developed by third-party developers and provide additional functionalities to the internet browser (in our case, *Firefox*). They enhance user experience by allowing customization and integration of various services. Some popular examples of *Firefox* add-ons include *AdBlocker*, which blocks intrusive advertisement, *Grammarly*, a tool for improving writing quality, and *LastPass*, a password manager. The creation of add-ons is accessible to a broad range of developers, ranging from established developers that have introduced multiple add-ons over time to newly entering developers. *Firefox* provides comprehensive developer documentation and toolkits to guide and facilitate the add-on creation process. The open-source nature of the platform encourages innovation, with a supportive community that contributes to forums and documentation pages. By focusing on *Firefox* add-ons, we are able to access the creation date of all available *Firefox* add-ons (or browser extensions) until June 2023, the full archive of available user reviews, and the last 25 versions before the current version.⁶

4.2 | Data

Our initial data set comprises 31,265 *Firefox* add-ons (i.e., software products). This dataset includes information on the release time, the last 25 versions of the add-ons, the creator(s), the add-on category, all available user reviews, and a snapshot of the number of active users. Since *Firefox* only provides information on the last 25 versions and there might have been more updates within the study's time frame (June 2021 to May 2023), we also collected additional data from archive.org. This approach allows us to capture all updates that occurred within this period and ensures comprehensiveness of the data. Furthermore, we also collected the number of active users over time by using archive.org. On *Firefox*, there are a total of 15 add-on categories, such as Appearance, Games & Entertainment, Language Support or Privacy & Security.⁷ While an add-on may span multiple categories, we consider the category listed first as the actual category in our analyses. Figures S1.1 and S1.2 in Supporting Information S1 show the distribution of add-ons across different categories.

In addition to our main dataset, we also collected primary data through a survey of 33 *Firefox* add-on developers. The survey, detailed in Appendix D, captures developers' experiences with and perceptions of GAI, including its effect on their ideation process, implementation workflow, development timelines, and competitive environment. While the sample size of the survey is relatively small,⁸ this mixed-methods approach allows us to complement our

⁶Note that, for other large browsers such as *Google Chrome*, *Safari*, or *Microsoft Edge*, such comprehensive information on add-ons is not publicly available.

⁷The full list includes: Alerts & Updates, Appearance, Bookmarks, Download Management, Feeds, News & Blogging, Games & Entertainment, Language Support, Other, Photos, Music & Videos, Privacy & Security, Search Tools, Shopping, Social & Communication, Tabs, and Web Development.

⁸The main reason for the relatively small sample size is that we focus on *Firefox* add-on developers (as opposed to other types of developers), which are the main actors in our study. Naturally, this comes at the expense of having fewer potential respondents. Given that the main objective of the survey was to obtain more in-depth insights on the potential mechanisms behind our findings, we opted for information depth over breadth and made sure that all our respondents were indeed *Firefox* add-on developers. Because the sample is relatively small, we interpret the survey as suggestive mechanism evidence that complements—rather than substitutes for—our econometric results.

large-scale quantitative findings with direct insights from the actors at the center of this technological shift.

4.3 | Measures

We consider two levels of analysis. First, to examine changes in the number and characteristics of original products introduced, we conduct analyses at the category level. Specifically, we consider the number of new add-ons introduced in each category, referred to as *New Add-ons Introduced* in a given month. This corresponds to new product introductions in a given market. Furthermore, we calculate the average number of categories (*Category Count*) that newly introduced add-ons span in a given month and use this as an dependent variable in our category-level model to measure the scope of knowledge domains that original software products draw on.⁹

Second, to measure product improvements (i.e., generational innovation), we use a variable counting the number of add-on updates, labeled as *Number of Updates*. Software products, exhibiting a high degree of technological flexibility, allow developers to refine, adapt, and improve them even after their initial market release. This attribute makes updates a valuable proxy for tracking generational innovation,¹⁰ and a valuable complement to new product introductions which measure original innovation.¹¹ Additionally, we employ a dummy variable (*Updated*), assigned a value of one if an update occurs in a given month and zero otherwise. Our analysis of add-on updates is conducted at the add-on-month level. For this purpose, we discard the add-ons introduced after June 2021 to align with the research design. Additionally, since we utilize add-on-treatment category fixed effects (as we explain below), we also eliminate add-ons that had not received updates during both the control and treatment periods.¹² This process resulted in a final count of 1,745 add-ons, yielding a total of 41,880 ($=1,745 \times 12 \times 2$) observations in our sample.

To circumvent the potential issue of “bad control”—which involves including controls that may be influenced by the treatment and could subsequently result in biased estimates of the treatment effect—we do not control for any other variables that might be affected by the treatment (Angrist & Pischke, 2009), such as review score, number of reviews or the number of active users. Table 1 report summary statistics and correlations. Supporting Information S1 reports additional summary statistics and correlations for each of our analyses, including a

⁹Note that the taxonomy of categories used by *Firefox* does not change. Our variable therefore reflects a (potentially) wider functional scope per product, not a proliferation of categories.

¹⁰Using updates as a measure of innovation aligns with existing literature that utilizes app updates to capture digital innovation in comparable contexts. For instance, Boudreau (2012) conducts an analysis of the number of application updates in handheld applications to understand innovation. Similarly, Tiwana (2015), Foerderer et al. (2018), and Wen and Zhu (2019) use updates as an indicator to infer the frequency and likelihood of innovation.

¹¹Previous literature has categorized new software products and major updates as “major innovations,” but also noted that launching a new product often requires more effort (Foerderer et al., 2018; Miric & Jeppesen, 2020; Wen & Zhu, 2019). Relatedly, these studies refer to minor updates as “incremental innovations” (Foerderer et al., 2018; Miric & Jeppesen, 2020; Wen & Zhu, 2019).

¹²Our fixed-effects regression requires within-unit variation in the outcome (updates) to identify an effect. Add-ons with no updates over the respective time period cannot contribute identifying variation—there is simply no change to model. If we did not remove them, such add-ons would be removed by the model. In Supporting Information S14, we show that alternative specifications lead to qualitatively similar results.



TABLE 1 Summary statistics and pairwise correlations for (a) category level analyses and (b) add-on level analyses.

Variables	Obs	Mean	Std. dev.	Min	Max	(1)	(2)	(3)
(1) New add-ons introduced	360	26.336	18.695	4	107	1.000		
(2) Log. new add-ons introduced	360	3.059	.643	1.386	4.673	0.932	1.000	
(3) Post $\times$ Treatment	360	.292	.455	0	1	0.231	0.247	1.000

Variables	Obs	Mean	Std. dev.	Min	Max	(1)	(2)	(3)	(4)
(1) Number of updates	41,880	.419	.958	0	24	1.000			
(2) Log. number of updates	41,880	.23	.429	0	3.219	0.938	1.000		
(3) Updated	41,880	.254	.436	0	1	0.749	0.919	1.000	
(4) Post $\times$ Treatment	41,880	.292	.455	0	1	−0.041	−0.042	−0.037	1.000

breakdown by treatment and control group before and after November, along with further descriptive tables and figures.

4.4 | Research design

Often, certain shocks impact all participants in the economy, making the establishment of a causal effect using observational data challenging. This is also the case for the introduction of *ChatGPT*, which affects different product categories and products simultaneously, making the selection of a suitable control group challenging. A viable approach to address this issue is to adopt a method called PD (Eichenbaum et al., 2020; Goldberg et al., 2024). Different from a difference-in-differences (DiD) design, where a contemporaneous untreated group absorbs time effects, PD has no such group and instead uses the same calendar months from the prior year as the control. In particular, PD proceeds via a twofold comparison. First, for the within comparison, we compare innovation activity during the months before (pre) and after (post) the introduction of *ChatGPT* in November 2022.¹³ Second, for the between comparison, we account for broader time trends by using the same calendar months from the previous year as a control group. The treatment group covers the period from June 2022 to May 2023, which is centered around the introduction of *ChatGPT* in November 2022. The control period (effectively serving as the control group¹⁴) consists of the corresponding months from the prior year, June 2021 to May 2022, providing a baseline for comparison across identical seasonal periods.¹⁵ Together, these comparisons help us distinguish changes associated with the introduction of *ChatGPT* from broader temporal patterns in innovation activity. This design capitalizes on the sharp

¹³In our baseline specification, November 2022 is classified as part of the post-treatment period, even though *ChatGPT* was released on November 30. This yields more conservative estimates; the results remain robust—and coefficients become larger—if November is assigned to the pre-treatment period.

¹⁴For simplicity, we use the terms “control period” and “control group” interchangeably.

¹⁵In Supporting Information S6, we report results from an alternative approach using different sets of control groups within a classical difference-in-differences analysis framework.

inter-temporal shock to innovation activity triggered by the release of *ChatGPT*; a significant event underscored by *ChatGPT*'s rapid global adoption by 100 million users within 2 months. Figure 1 provides an overview of the research design.

This strategy hinges on three assumptions for identifying a causal effect. First, the parallel-trends assumption: in the absence of the treatment (i.e., the launch of *ChatGPT*), the difference in the dependent variable between the 2022/2023 period and the 2021/2022 period should have remained the same in the post-period (November to May) as it was in the pre-period (June to October).¹⁶ Importantly, this is a statement about trends rather than levels: even if absolute add-on activity is somewhat higher or lower in 2021/2022 than in 2022/2023, the assumption is satisfied as long as the two periods follow a parallel trajectory over time (in the absence of treatment). Similar to the DiD design, it is not possible to directly test the parallel-trends assumption, as it is not possible to observe untreated units in the treatment group after the treatment. What can be tested are the pre-trends in the months before *ChatGPT*'s introduction (June through October), which reflect whether the 2022/2023 and 2021/2022 series moved in parallel before November. These pre-trend analyses, which we report in Figures A1 and A2 in Appendix A, as well as the additional analyses conducted in Supporting Information S5, provide evidence that is consistent with the validity of the parallel-trends assumption.

One implication of the PD framework is worth making explicit in this regard. In DiD, the parallel-trends assumption requires that the treated and control (absent treatment) would have behaved similarly *over the same time period*. In PD, it is required that the treated and control

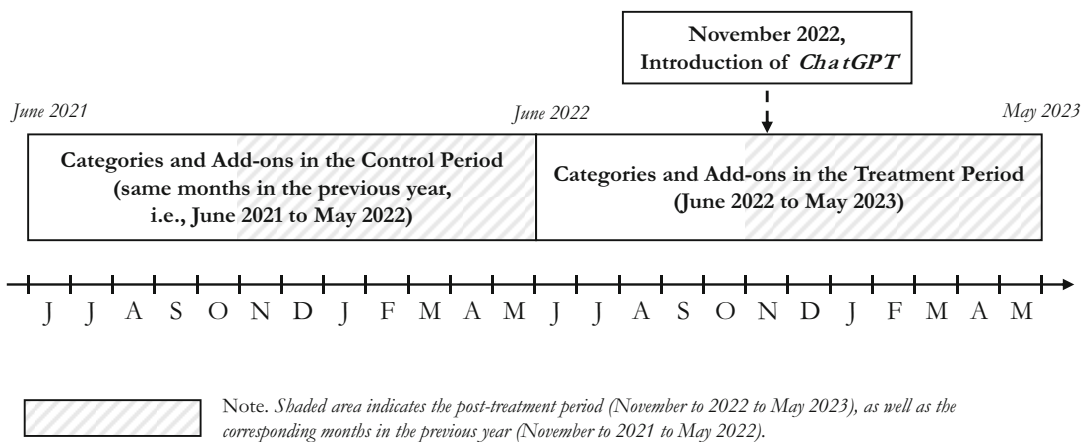


FIGURE 1 Research design. This figure illustrates the research design implemented in this study. Our analysis covers the time period from June 2021 through May 2023. We use the period from June 2022 through May 2023 as the treatment period and the corresponding months in the previous year (i.e., June 2021 through May 2022) as the control period. The shaded areas indicate the post-treatment period (November 2022 to May 2023), as well as the corresponding months in the previous year (November 2021 to May 2022). The panel-differences estimator contrasts the months from November onward with the months pre-November within each year and then differences these contrasts across the 2 years.

¹⁶Equivalently, the two periods should have continued to evolve in parallel after November 30, 2022, just as they did before.



(absent treatment) would have behaved similarly *across different time periods* (here, 2021/2022 and 2022/2023, with *ChatGPT* introduced in November 2022). The latter is a stronger assumption, as it presumes year-over-year stability in the data-generating process beyond the seasonality captured by month-of-year fixed effects.

A direct implication of the parallel-trends assumption is that no event other than the introduction of *ChatGPT* should asymmetrically affect outcomes across the two periods. The most plausible candidate in our setting is browser-level updates during the treatment period that could inflate add-on updates. Therefore, we replicate our main analyses excluding the add-on categories most likely to be affected and find that our results hold (Supporting Information S3).¹⁷

Second, the treatment must be exogenous with respect to the *Firefox* add-on ecosystem. This condition is satisfied as the introduction of *ChatGPT* happened independently of the *Firefox* ecosystem and was not prompted by changes in the *Firefox* ecosystem. In the PD framework, this assumption is what allows us to treat the launch of *ChatGPT* in November 2022 as an as-if-random temporal shock.

Third, the stable unit treatment value assumption must be satisfied, ensuring the treatment is qualitatively similar for all units and no interference occurs between the treatment and control groups. This condition is met as the treatment has no variations, and temporal separation between the groups eliminates interference.

4.5 | Empirical model

In our analyses, we employ ordinary least squares (OLS), log-OLS, and Poisson models for category-level analyses. For add-on level analyses, we utilize OLS, log-OLS, Poisson, and Logit models.¹⁸ Specifically, we employ the following general regression specification:

$$Y_{it} = \alpha + \beta_1 \times \text{Post} \times \text{Treatment}_{it} + \gamma_i + \delta_t + \epsilon_{it}$$

where γ_i is the category-treatment group or add-on-treatment group.¹⁹ In this context, the “treatment group” differentiates whether the observations of the category or add-on originate

¹⁷We conducted interviews with developers of *Firefox* add-ons to identify potential browser-level updates that may interfere with our findings, as well as the types of apps that are most likely to be affected by these updates. A more detailed explanation of this procedure can be found in Supporting Information S3.

¹⁸We believe Poisson is the most appropriate model for several reasons. First, add-ons are count data (0, 1, 2, and so on), which Poisson models are designed to handle. Second, Poisson models naturally accommodate zero outcomes without requiring ad hoc remedies such as dropping or arbitrarily shifting zeros, unlike log-linear OLS (Chen & Roth, 2024). Third, Poisson models also remain robust under various heteroskedastic patterns and rounding-error scenarios (Silva & Tenreyro, 2006), yield consistent estimates regardless of whether the dependent variable is integer-valued, continuous, or count (Cameron & Trivedi, 2013; Gourieroux et al., 1984; Powell & Seabury, 2018; Silva & Tenreyro, 2006), and uniquely enable the inclusion of high-dimensional fixed-effects without bias, unlike fixed-effects negative-binomial models (Allison & Waterman, 2002; Martin & Pham, 2020; Silva & Tenreyro, 2011). Nonetheless, we include the OLS and log-linear models alongside Poisson to show that our results are robust across different specifications.

¹⁹We have 15 categories over 12 months in each period (control and treatment, i.e., 2 years). This gives us $15 \times 12 \times 2 = 360$ observations, 180 each in the treatment and control groups. In the category-level models, we use “category $\times$ treatment-group” fixed-effects. We have 30 fixed-effects, corresponding to the 30 “category-treatment groups” (15 categories $\times$ 2 [treatment and control]). We observe each of these category-treatment groups over 5 months pre and 7 months post ($30 \times [5 + 7] = 360$).

from the treatment year (that is, the year of the introduction of *ChatGPT* in November 2022, with observations gathered before and after this event) or from the control year (which refers to the year preceding the introduction of *ChatGPT*, serving as a baseline for comparison).²⁰ δ_t are month-of-the-year fixed effects controlling for innovation activity in certain months of the year, that is, seasonality, such as holiday period. Y_{it} indicates any dependent variable. The term $Post \times Treatment$ is one for observations after the introduction of *ChatGPT*, zero otherwise. It is important to note that this estimation does not suffer from staggered adoption as there is a single treatment date for all the categories or add-ons.

5 | RESULTS

In this section, we present the results of our analyses. We test our hypotheses and conduct a series of additional analyses that shed light on potential mechanisms driving our main findings. These additional findings are based on both our main dataset and the results from an exploratory survey of 33 *Firefox* add-on developers (see Appendix D for an overview of survey results).

5.1 | Add-ons introduced

Figure 2 shows the number of new add-ons introduced (by existing or newly entering developers) in the 2022/2023 and the 2021/2022 period. It can be seen that, initially, the trends in the number of add-ons introduced follow a similar pattern compared to the control group. However, after the introduction of *ChatGPT*, there is an increase in the number of new add-ons introduced.

We report the results from PD analyses at the category level in Table 2. In column 1, the $Post \times Treatment$ coefficient of 7.366 ($p = .001$) means that, on average, the introduction of *ChatGPT* in the post-treatment period is associated with an increase of approximately 7.366 new add-ons being introduced. The $Post \times Treatment$ coefficient in column (2) of 0.295 ($p = .0003$) suggests that, after the treatment, the logarithm of the number of new add-ons introduced increases by 0.295, on average. This corresponds approximately to a 34.4% increase in the number of new add-ons ($\exp(0.295) - 1 \approx 0.344$) after the treatment. The $Post \times Treatment$ coefficient in column 3 is 0.248 ($p < .0001$), indicating that the logarithm of the expected count of new add-ons introduced is expected to increase by 0.248 after the treatment. Exponentiating this coefficient ($\exp(0.248) - 1 \approx 0.282$) indicates approximately a 28.2% increase in the expected count of new add-ons after the treatment. Taken together, these results are consistent with Hypothesis 1. In Figure A1 in Appendix A, we plot the leads and lags of the treatment. It can be seen that the pre-treatment coefficients are close to zero (also confirmed by an F -test; $F = 0.05$, $p = .82$), and the post-treatment coefficients are substantially larger than zero, which lends credibility to the parallel trends assumption.

Survey responses are consistent with the mechanism that GAI facilitates both ideation and implementation (see Appendix D). Many respondents report that GAI plays a dual role in

²⁰In *Stata*, we created this data structure by grouping observations at the treatment-category or treatment-add-on level, that is, `egen new_id = group(category, treatment)` or `new_id = group(addon_id, treatment)`. An alternative to this specification would be to add “Treatment” as a separate regressor, and add category fixed-effect instead of “category $\times$ treatment-group” fixed-effects. This alternative approach is reported in Supporting Information S14.

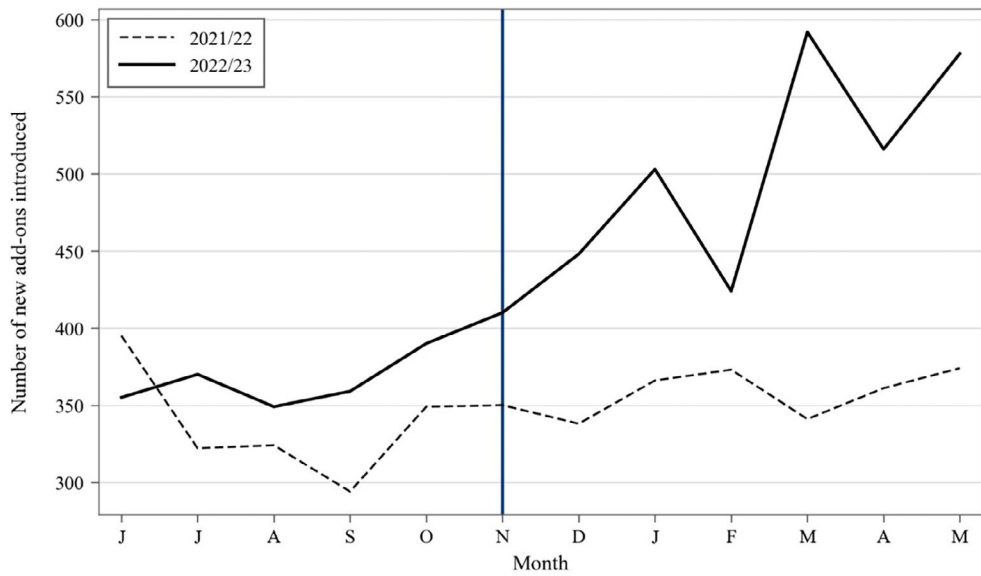


FIGURE 2 Number of new add-ons introduced. This figure plots the number of new add-ons introduced to the *Mozilla Firefox* ecosystem in each calendar month from June through May. The dashed 2021/2022 line represents the period from June 2021 to May 2022; the solid 2022/2023 line represents the period from June 2022 to May 2023. The vertical line marks the introduction of *ChatGPT* in November 2022 and the corresponding month in the previous year.

TABLE 2 Number of new add-ons introduced.

	(1) New add-ons introduced	(2) Log. new add-ons introduced	(3) New add-ons introduced
Post × Treatment	7.3657 (2.0216) [0.0010]	0.2950 (0.0727) [0.0003]	0.2476 (0.0524) [0.0000]
Constant	24.1878 (0.5896) [0.0000]	2.9734 (0.0212) [0.0000]	3.3724 (0.0192) [0.0000]
Category-treatment FE	Yes	Yes	Yes
Month FE	Yes	Yes	Yes
Adjusted R^2	0.883	0.832	
Pseudo R^2			0.628
Log-likelihood	−1172.0	−23.06	−2899.1
Observations	360	360	360

Note: Unit of observation: *Category(i)*. Models: *OLS*; *Log-OLS*; *Poisson*. Standard errors are robust to heteroskedasticity, clustered at the category-treatment group pair level and reported in parentheses. *p*-Values are reported in brackets.

affecting both ideation and implementation. Regarding ideation, a strong majority of respondents agreed that GAI helps them generate new ideas and overcome creative blocks. For implementation, the results were even more pronounced. Respondents overwhelmingly agreed that

GAI helps them write code faster and, critically, develop functionalities they previously lacked the skills to create. This is reflected in a dramatic self-reported reduction in project completion times from a scale of months to weeks after GAI became available, providing strong support for the immediacy of the effects we document.

5.2 | Add-ons introduced by new developers

Table 3 shows the results from our analysis of new add-ons introduced by *new* developers (columns (1) to (3)), as well as from a comparison between existing and new developers²¹ (columns (4) to (6)) following the introduction of *ChatGPT*.²² The first three models reveal a positive impact on the number of new add-ons introduced by new developers during the post-treatment period. Column (1) shows that the *Post* $\times$ *Treatment* coefficient is 5.964 ($p = .0020$), indicating that new developers, on average, introduced approximately 5.964 more new add-ons following the introduction of *ChatGPT*. The result is similar when using the Log-OLS (column (2)) and Poisson (column (3)) regression models. This provides evidence in support of Hypothesis 2. Columns (4) to (6) show the outcomes for new developers in comparison to existing ones. The *Post* $\times$ *Treatment* coefficients in columns (4) to (6) again show a substantial increase in new add-ons introduced post-treatment by new developers (e.g., column (4): $\beta = 6.262$, $p = .0242$). At the same time, the interaction term (*Post* $\times$ *Treatment* $\times$ *Existing*) in columns (4) to (6) does not provide any evidence that the increase in the number of new add-ons introduced by new and existing developers differs in the post-treatment period (e.g., column (4): $\beta = -5.1591$, $p = .1519$). Hence, while entry barriers after the availability of *ChatGPT* seem to be lower, the increase in add-ons introduced by new developers appears to occur along with an increase in add-ons introduced by existing developers.

Exploratory survey evidence also suggests a channel through which entry increases (see Appendix D). Less-experienced developers reported particularly strong gains from using GAI to translate high-level objectives into working code, which can compensate for limited experience. Specifically, they reported using GAI to “program add-on functionalities that [they] previously lacked the skills to develop.” Furthermore, when asked if GAI made it easier to enter the add-on market, a majority of developers (69.7%) agreed, with less experienced developers showing significantly stronger agreement. As one new developer stated, “I learnt everything related to programming from *ChatGPT*,” highlighting GAI’s role in lowering entry barriers.

5.2.1 | Quality of add-ons by new developers

A natural concern is that add-ons introduced by new developers are of lower quality than those of existing developers. This might be the case, for instance, because the lack of experience of new developers cannot be fully compensated by GAI assistance or, more generally, because the availability of GAI assistance leads to a sudden increase in *perceived* ability of developers with limited ability. To investigate this further, we look at the ratings that new add-ons receive. Specifically, we create the variable *Average Rating*, which reflects the average rating of an add-on at the time of data collection and the variable *Average Rating 30 Days*, which reflects the average rating of an add-on within 30 days after its release. We aggregate this data at the category-

²¹An add-on is considered an add-on by a new developer if the focal developer has not introduced any add-ons before.

²²Summary statistics for these analyses are provided in Supporting Information S1 in Table S1.1 and Table S1.2.



TABLE 3 Number of new add-ons by new versus existing developers.

	(1) New add-ons introduced	(2) Log. new add-ons introduced	(3) New add-ons introduced	(4) New add-ons introduced	(5) Log. new add-ons introduced	(6) New add-ons introduced
Post $\times$ Treatment	5.9638 (1.7516) [0.0020]	0.2471 (0.0684) [0.0011]	0.2547 (0.0544) [0.0000]	6.2624 (2.6330) [0.0242] −14.2980 (1.9410) [0.0000] −5.1591 (3.5059) [0.1519]	0.2198 (0.0816) [0.0116] −1.3043 (0.0358) [0.0000] 0.0431 (0.0737) [0.5632]	0.2363 (0.0555) [0.0000] −1.4072 (0.0376) [0.0000] 0.0562 (0.0656) [0.3913]
Existing						
Post $\times$ Treatment $\times$ Existing						
Constant	19.3300 (0.5109) [0.0000]	2.8254 (0.0200) [0.0000]	3.1532 (0.0198) [0.0000]	19.2429 (1.0169) [0.0000]	2.8334 (0.0286) [0.0000]	3.1535 (0.0203) [0.0000]
Category-treatment FE	Yes	Yes	Yes	Yes	Yes	Yes
Month FE	Yes	Yes	Yes	Yes	Yes	Yes
Sample	New developers	New developers	New developers	All developers	All developers	All developers
Adjusted R^2	0.877	0.822		0.736	0.818	
Pseudo R^2			0.596			0.663
Log-likelihood	−1107.8	−12.51	−2520.8	−2718.9	−816.9	−5508.7
Observations	360	360	360	720	720	720

Note: Unit of observation: *Category(i)*. Models: *OLS (1,4)*; *Log-OLS (2,5)*; *Poisson (3,6)*. Standard errors are robust to heteroskedasticity, clustered at the category-treatment group pair level and reported in parentheses. *p*-Values are reported in brackets. Columns (1) to (3) show the results for new developers. Columns (4) to (6) compare existing developers with new developers.

treatment level, and we again employ an OLS (1,3,5,7) and Log-OLS (2,4,6,8) model to study the effect of the introduction of *ChatGPT* on new (1,2,5,6) and existing versus new (3,4,7,8) developers.

The *Post* $\times$ *Treatment* coefficient in columns (1) through (8) of Table 4 carries a negative sign but provides no evidence that the average quality of add-ons by new developers has decreased after the availability of GAI assistance (e.g., column (1): $\beta = -0.1027$, $p = .5090$). Moreover, the coefficient of the three-way interaction term *Post* $\times$ *Treatment* $\times$ *Existing* shows no evidence that the quality of add-ons by new developers has decreased compared to that of existing developers (e.g., column (4): $\beta = -0.0872$, $p = .4192$). In fact, in one of the specifications (i.e., column (3): $\beta = -0.2507$, $p = .0949$), there is even *some* limited evidence that the quality of add-ons by existing developers may have decreased compared to new developers. Together, these results thus do not provide any evidence that the availability of GAI assistance has led to a quality decrease of add-ons by new developers.

5.3 | Updates introduced

Table 5 contains the results from our add-on level analysis of the number of updates (i.e., improvements of existing products).²³ It can be seen from Table 5 that the introduction of *ChatGPT* has a negative effect on the number of updates introduced. In column (1), the *Post* $\times$ *Treatment* coefficient of -0.0909 ($p < .001$) suggests that, on average, the introduction of *ChatGPT* in the post-treatment period is associated with a decrease of approximately 0.0909 updates being introduced per add-on. The *Post* $\times$ *Treatment* coefficient in column (2) is -0.0481 ($p < .001$), indicating that after the treatment, the logarithm of the number of updates decreases by 0.048 on average. This represents a decrease of approximately 4.69% in the number of updates ($\exp(-0.0481) - 1 \approx -0.0469$) after the introduction of *ChatGPT*. The pattern of results remains similar in columns (3) and (4), where we use a Poisson and Logit model, respectively. For instance, the *Post* $\times$ *Treatment* coefficient of -0.226 ($p < .001$) indicates that the logarithm of the expected count of updates decreases by 0.226 following the treatment, which corresponds to a 20.3% ($\exp(-0.226) - 1 \approx -0.203$) decrease in the expected count of updates per add-on following the introduction of *ChatGPT*. Taken together, this provides support for Hypothesis 3. In Figure A2 in Appendix A, we again plot the leads and lags of the treatment. The pre-treatment coefficients are close to zero ($F = 0.26$; $p = .6129$), and the post-treatment coefficients are significantly smaller than zero, lending credibility to the parallel trends assumption and supporting the observed negative effect of *ChatGPT* on the number of add-on updates.

5.4 | Motivations for the shift toward original innovation (and away from product improvements)

To gain additional insights into why developers seem to be reallocating resources from product improvements to original innovation, we draw on qualitative evidence from add-on developers. While the developers did not explicitly state that they are making fewer updates than before, they all reported using *ChatGPT* for their new projects. Below are relevant quotes:

²³Figure S1.3 in Supporting Information S1 shows the number of new add-on updates introduced in the 2022/2023 and the 2021/2022 period.



TABLE 4 Quality of add-ons by existing versus new developers.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	Average rating	Log. average rating	Average rating	Log. average rating	Average rating 30 days	Log. Avg. rating 30 days	Average rating 30 days	Log. Avg. rating 30 days
Post $\times$ Treatment	-0.1027 (0.1536) [0.5090]	-0.1554 (0.1098) [0.1678]	-0.0588 (0.1668) [0.7268]	-0.1779 (0.1121) [0.1235]	-0.2641 (0.1600) [0.1096]	-0.0734 (0.0502) [0.1547]	-0.1908 (0.1358) [0.1709]	-0.0510 (0.0429) [0.2440]
Existing		0.0508 (0.1010) [0.6187]	0.0508 (0.1010) [0.6187]	0.2227 (0.0595) [0.0008]			-0.0984 (0.1022) [0.3436]	-0.0336 (0.0359) [0.3569]
Post $\times$ Treatment $\times$ Existing			-0.2507 (0.1452) [0.0949]	-0.0872 (0.1064) [0.4192]			-0.0425 (0.1503) [0.7795]	-0.0181 (0.0523) [0.7310]
Constant	1.4910 (0.0448) [0.0000]	0.3326 (0.0314) [0.0000]	1.4780 (0.0569) [0.0000]	0.3472 (0.0312) [0.0000]	4.8375 (0.0476) [0.0000]	1.5691 (0.0150) [0.0000]	4.8099 (0.0520) [0.0000]	1.5610 (0.0165) [0.0000]
Category-treatment FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sample	New developers	New developers	All developers	All developers	New developers	New developers	All developers	All developers
Adjusted R^2	0.188	0.228	0.113	0.184	0.045	0.026	0.046	0.038
Log-likelihood	-323.0	-210.5	-964.6	-453.3	-256.8	108.3	-465.1	62.15
Observations	360	350	701	597	329	329	489	489

Note: Unit of observation: *Category(i)*. Models: *OLS* (1,3,5,7); *Log-OLS* (2,4,6,8). Standard errors are robust to heteroskedasticity, clustered at the category-treatment group pair level and reported in parentheses. *p*-Values are reported in brackets. Models 1, 2, 5 and 6 show the results for new developers. Models 3, 4, 7, and 8 compare existing developers with new developers. The sample sizes in the models differ because not all category-treatment clusters contain add-ons that received at least one rating within the specified time frame.

TABLE 5 Number of new updates introduced.

	(1) Number of updates	(2) Logarithm of number of updates	(3) Number of updates	(4) Updated (0/1)
Post $\times$ Treatment	−0.0909 (0.0215) [0.0000]	−0.0481 (0.0095) [0.0000]	−0.2263 (0.0505) [0.0000]	−0.3233 (0.0494) [0.0000]
Constant	0.4456 (0.0063) [0.0000]	0.2442 (0.0028) [0.0000]	−0.3365 (0.0126) [0.0000]	
Add-on-Treatment FE	Yes	Yes	Yes	Yes
Month FE	Yes	Yes	Yes	Yes
Adjusted R^2	0.183	0.194		
Pseudo R^2			0.210	0.003
Log-likelihood	−51586.6	−17649.8	−39768.8	−15041.8
Observations	41,880	41,880	41,880	41,700

Note: Unit of observation: *Add-on(i)*. Models: *OLS*; *Log-OLS*; *Poisson*; *Logit*. Standard errors are robust to heteroskedasticity, clustered at the add-on-treatment group pair level and reported in parentheses. *p*-Values are reported in brackets.

Interviewee 1: “... Well, I’ve used ChatGPT as an opportunity to experiment with some of my ideas. I honestly find it quite efficient in implementing my ideas and debugging code. I’ve also used ChatGPT to generate some UI design ideas. [...] I mean, I also used ChatGPT to improve my extensions here and there, but I spent more time on creating new add-ons. I was also somewhat concerned that others might use ChatGPT and do all the cool stuff before I do.”

Interviewee 2: “...Unless you’ve made an add-on for the privacy category, or download management, which require frequent updates, you don’t really lose much if you don’t make any updates. [...] Also, I don’t think using ChatGPT to update my code would be efficient since it would require teaching my scripts to ChatGPT. ChatGPT is quite good at CSS (referring to Cascading Style Sheets), and I use it to create theme extensions with much less effort.”

Interviewee 3: “... To be honest, ChatGPT encouraged me to pursue my other ideas, outside of the extension business, some of which are also related to ChatGPT. I made last major update to my extension in February (referring to 2023) and now I am focusing on other projects.”

In line with our theoretical reasoning, developers seem to see opportunities for generating novel ideas with the help of GAI assistance, but see more limited use for GAI assistance in the context of product improvements, which ultimately drives them to reallocate their effort accordingly. These results are corroborated by other insights from our survey (see Appendix D). Developers strongly agreed that the availability of GAI has led them to work more on *new* add-on projects ($M = 6.09$, $SD = 1.15$) and that it has freed up resources to pursue new ideas ($M = 6.36$, $SD = 0.96$). This shift is further explained by perceived market dynamics (see Figure D8).



Developers feel a greater need to develop *new* add-ons to stay competitive ($M = 5.76$, $SD = 1.15$), driven by a worry that others will be faster in both conceiving ($M = 5.52$, $SD = 1.70$) and implementing ($M = 5.91$, $SD = 1.33$) new ideas.

5.5 | Scope of knowledge domains in original software products

As we discussed above, we expect that, once GAI assistance becomes available, software products will draw on a broader scope of knowledge domains. To understand if this is the case, we analyze whether newly introduced add-ons span more add-on categories after the introduction of *ChatGPT*. The results are reported in Table 6, with column (1) presenting the results using the average *Category Count*. We find that, after the introduction of *ChatGPT*, the average number of categories that add-ons span increases ($\beta = 0.057$, $p = .0241$), which is consistent with Hypothesis 4.

To analyze whether new add-ons are *novel* (i.e., different from add-ons that were previously introduced to the market), we use the textual similarity of descriptions of newly introduced add-ons. If add-ons are indeed novel, we should see that their descriptions are significantly different from (i.e., less similar to) previous ones. We use *Latent Semantic Indexing (LSI) similarity scores*²⁴ for the textual similarity of add-on descriptions and replicate our PD analysis at the add-on category level.²⁵ Column (2) presents the results using the *LSI Similarity Score* for the original description text, and column (3) shows the results with the *LSI Similarity Score* for

TABLE 6 Novelty of new add-ons introduced.

	(1) Category count	(2) LSI similarity score	(3) LSI similarity score (neutralized text)
Post $\times$ Treatment	0.0568 (0.0239) [0.0241]	0.0070 (0.0042) [0.1108]	0.0069 (0.0032) [0.0416]
Constant	1.3700 (0.0070) [0.0000]	0.1514 (0.0012) [0.0000]	0.1989 (0.0009) [0.0000]
Category-treatment FE	Yes	Yes	Yes
Month FE	Yes	Yes	Yes
Adjusted R^2	0.811	0.128	0.242
Log-likelihood	312.2	957.1	1009.4
Observations	360	360	360

Note: Unit of observation: *Category(i)*. Model: OLS. Standard errors are robust to heteroskedasticity, clustered at the category-treatment group pair level and reported in parentheses. *p*-values are reported in brackets.

²⁴See additional explanations in Supporting Information S2. Results remain unchanged with *TF-IDF-based Scores* or *Semantic Embeddings*.
²⁵Summary statistics for these analyses are reported in Table S1.4 in Supporting Information S1.

the neutralized description text (which was adjusted to reduce potential style differences resulting from GAI-assisted text generation²⁶) as the dependent variable. For both measures, we do not find any evidence suggesting that GAI assistance results in more novel (i.e., less similar) add-ons being developed (e.g., column (2): $\beta = 0.0070$, $p = .1108$). In fact, in one of our specifications (column (3)), we even find *some* evidence that products become more similar to previous products. Hence, while developers may combine knowledge elements from different domains, this may not necessarily result in more novel add-ons. Our findings are thus consistent with the idea of “efficient recombination”, that is, developers becoming more efficient at generating ideas that are based on new combinations of knowledge but are still relatively similar to previous ones.

5.6 | Robustness checks and additional analyses

We also conduct additional robustness checks to ensure the integrity of our results. We briefly describe them here and provide more details in the Appendix and Supporting Information.

5.6.1 | Alternative econometric approaches

First, we utilize the synthetic difference-in-differences (SDID) approach, which combines elements of the Synthetic Control method and DiD to reduce reliance on the parallel trends assumption and improve the robustness and precision of the estimator (Arkhangelsky et al., 2021) (Supporting Information S5). Second, we conduct interviews with developers to identify the categories most positively influenced by *ChatGPT* and employ a classical DiD approach comparing these categories to others (Supporting Information S6). Third, we incorporate category- and add-on-specific time trends to control for the possibility that different categories or add-ons might follow distinct trends (Supporting Information S7). Fourth, we verify that our findings are robust to using a standard DiD framework and to alternative fixed-effects structures to ensure our results are not driven by model choice (Supporting Information S14).

5.6.2 | Ruling out alternative explanations

First, the increase in new add-ons could reflect excitement about AI-themed opportunities rather than changes on the production (i.e., coding) side. To account for potential demand-side effects triggered by the introduction of *ChatGPT*, we exclude all AI-related add-ons from the sample (Appendix C). Second, to address potential longevity and update frequency issues in software product lifecycles, we narrow the sample to products no more than 3 months old in the pre-treatment period for both control and treatment groups (Supporting Information S8). Third, we replicate our main analysis with a subsample that excludes professional (as opposed to non-professional) developers to account for potential differences in their incentives to develop original add-ons or updates (Supporting Information S9). Fourth, we use different specifications to measure add-on novelty, including (“neutralized”) TF-IDF, LSI, and semantic embeddings scores to confirm that our novelty results are not driven by methodological choices

²⁶See Supporting Information S2, Section S2.2 for additional information.



(Supporting Information S2). Fifth, we examine whether our findings might instead reflect a reduced need for collaboration among developers, and find no evidence that the number of developers per new add-on declines (Supporting Information S13).

5.6.3 | Additional evidence for our proposed mechanisms

First, we test whether competitive pressure from new entrants moderates the shift toward original innovation (Supporting Information S12). Second, we measure changes in the number of new developers and total developers over time to assess whether lowered entry barriers are a plausible mechanism (Supporting Information S15). Third, we use an AI detection tool to check the source code of browser add-ons for GAI-generated content, which provides evidence that developers do indeed use GAI tools post-introduction (Supporting Information S10). Fourth, we look at the differential effect of *ChatGPT* on major and minor updates and find that, while it did not alter the number of major updates introduced, it did result in a reduction of minor updates (see Appendix B). Fifth, we find that the decrease in the number of updates is less pronounced for add-ons with larger user bases, which is consistent with the (re)allocation of effort between original and generational innovation being driven, at least in part, by developer incentives (see Appendix B). Sixth, we disaggregate the treatment effect across product categories and find a broadly consistent pattern of positive effects (Supporting Information S4). Seventh, we show at the developer level that the number of add-ons a developer works on in parallel increases in the post-treatment period (Supporting Information S11). Lastly, we complement these quantitative analyses with evidence from our *Firefox* developer survey, which provides support for our proposed ideation and implementation mechanisms and helps rule out competing explanations (Appendix D). In particular, the survey illustrates how GAI does not merely act as an efficiency-enhancing tool, but fundamentally changes innovation patterns.

Across all these analyses, the results remain consistent with our main findings and with our theoretical reasoning.

6 | DISCUSSION

In this paper, we study the impact of GAI tools on product-level innovation outcomes in the context of software products. We argue that GAI's ability to assist developers in both the ideation and the implementation of products can enable some developers to create software products that were previously beyond their reach, while allowing others to reallocate their effort with respect to different types of activities. This, in turn, not only increases the rate of innovation but also affects the types of products that developers create, and ultimately the overall direction of innovation.

We use the introduction of *ChatGPT* into the market to study its effects on the *Firefox* add-on ecosystem. We find that the 34.4% surge in the introduction of new add-ons in the wake of *ChatGPT*'s entry stems not only from more add-ons developed by existing developers but also from more add-ons being introduced by new developers, thus suggesting that GAI assistance can broaden the sources of innovation. While original add-ons tend to span more different categories after the introduction of *ChatGPT* than before, this does not necessarily translate into more novel add-ons, as further analyses indicate. Consistent with our reasoning, the increase in new add-ons being introduced is contrasted by a 20.3% reduction in the rate of add-on updates.

Additional data gathered from a survey with 33 *Firefox* developers (see Appendix D) provides support for the theoretical mechanisms we describe.

6.1 | Contribution to the AI literature

Previous studies have largely focused on PAI, demonstrating for instance its ability to accelerate drug discovery (Lou & Wu, 2021), improve decision-making (Gaessler & Piezunka, 2023) or affect productivity (Acemoglu & Restrepo, 2020; Agrawal, Gans, & Goldfarb, 2019; Yilmaz, Naumovska, & Aggarwal, 2023). Studies that specifically explore the effect of GAI tools have mostly focused on whether they are complements or substitutes for human individuals in specific tasks (Boussieux et al., 2024; Brynjolfsson et al., 2025; Doshi & Hauser, 2024; Girotra et al., 2023; Noy & Zhang, 2023). What has remained unexplored however is how, as a reaction to complementarity or substitution effects, individuals may reallocate their effort with respect to different innovation activities and how this translates into a shift in the type of products that are developed and released into the market. With this paper, we address precisely this question and show that GAI drives individuals to shift their efforts toward new types of innovation activities, which in turn affects the trajectory of innovation. This allows us to shed light on how task-level effects cascade into a shift in the direction of innovation. Our detailed data on (commercializable) real-world software products allows us to extend previous research that has typically focused on *individual* elements in the innovation process (e.g., Boussieux et al., 2024; Doshi & Hauser, 2024) by observing potential changes in product outcomes and tracking changes in the trajectory of innovation.

More broadly, our results contribute to the ongoing debate about whether AI will primarily support exploitation of existing knowledge or enable more exploratory innovation (Agrawal et al., 2022). While our findings suggest that add-ons introduced after the release of *ChatGPT* span more add-on categories, and may therefore combine different types of knowledge, we do not find evidence that these add-ons are indeed less similar to previously introduced add-ons. These findings speak to theories that emphasize the positive association between knowledge recombination and innovativeness (Fleming, 2001; Katila & Ahuja, 2002). Our results indicate that GAI might be enabling a form of “efficient recombination”, where developers can more efficiently access and combine knowledge from diverse domains, rather than fundamentally altering the novelty of the ideas generated. This refines our understanding of how knowledge recombination translates to innovation outcomes in the age of AI, and calls for closer examination of the mechanisms through which diverse knowledge inputs contribute to innovation novelty. Moreover, this observation aligns with and extends recent work on human-AI collaboration in creative tasks (He et al., 2024; Jia et al., 2024), suggesting that while GAI can broaden the search space for ideas, the ultimate novelty of the output may still be bound by human factors such as domain expertise or creative vision.

6.2 | Contribution to the literature on innovation and general-purpose technologies

We contribute to the literature on how new technologies affect innovation trajectories and, more specifically, to existing research on general-purpose technologies and innovation (e.g., Bresnahan & Trajtenberg, 1995; Helpman, 1998). We show that the availability of GAI can



have an immediate impact on innovation and, more specifically, on the direction of innovation. This contrasts with previous research, which has found that the effects of prior general-purpose technologies were often delayed due to coordination problems between the GPT and its application sectors (Bresnahan & Trajtenberg, 1995). We attribute this difference to the fact that deep learning—and GAI in particular—is not only a GPT (Eloundou et al., 2024) but also an invention of a method of inventing (Cockburn et al., 2018). Specifically, while previous general-purpose technologies such as the steam engine may have led to innovation in the focal technology (i.e., the steam engine) and its application sectors (e.g., manufacturing machinery powered by steam engines), in the context of GAI, the technology (i.e., GAI) itself creates innovations by supporting the ideation and implementation phases of innovation. This unique characteristic of GAI may also explain the difference in its effect compared to other enabling technologies with more basic functionalities, such as providing standardized product components, which often lead to a decrease in product novelty after their introduction (Chen, Yi, et al., 2022; Miric et al., 2023).

Our results also speak to a growing stream of literature that studies how technologies that assist the innovation process can significantly affect the overall direction of innovation. These studies have mostly focused on technologies that facilitate data-based predictions (Agrawal et al., 2024; Tranchero, 2024) or automation (Furman & Teodoridis, 2020) in the innovation process, and have shown that they may accelerate innovation and increase innovation success. We complement these studies by analyzing how GAI, with its unique characteristics, may impact the types of products that individuals create and thus the direction of innovation, as well as how this is driven by new and existing developers.

6.3 | Generalizability to other innovation settings

Our study focuses on digital innovation (i.e., software product development) given its inherent characteristics, which provide a conducive environment to study the impact of GAI. In particular, digital innovation is characterized by the close relationship between GAI outputs and digital products, and allows for clear traceability of different types of software products and their updates. While our findings can also be extended to other settings, this requires a consideration of different industry characteristics.

In general, GAI's effect on the *first* phase of the innovation process is arguably more easily transferrable, as the capabilities of GAI tools in ideation and creativity could significantly contribute to innovation beyond the digital realm. In principle, we expect the basic mechanism of efficient recombination to expand to settings where there is a large body of knowledge based on which GAI can generate ideas. This is true, for instance, for other digital contexts, such as app development or video games, but also for scientific discovery where knowledge is codified and modular. In such settings, GAI can be helpful in accessing a broader set of knowledge elements and combinations.

Importantly however, while the basic mechanism of efficient recombination may be present in such settings, there may be additional factors determining whether this translates into actual product outcomes. For instance, more complex software projects typically involve a larger group of developers (with a hierarchical structure), as well as even stronger financial considerations. It is therefore unclear if our findings would scale linearly if project size and complexity increase. Nonetheless, even in large software projects, individual developers with limited experience may fill their skill gaps with GAI assistance. Similarly, while individual developers in these projects

may not independently decide how to allocate their efforts, it is still possible that a shift toward a higher number of original innovations can be observed at the aggregate level of the software project. Similarly, in highly regulated industries like pharmaceuticals, where innovation is constrained by safety requirements and extensive testing, the ability of GAI to dramatically alter innovation outcomes may be limited. In such contexts, while GAI might still enhance ideation, its impact on innovation patterns might be more modest.

Context-specific factors may also affect whether innovation outcomes are *original* but not *truly novel*. In areas such as drug-target identification or materials science, GAI may reduce the cost of searching and combining knowledge elements. At the same time, domain expertise remains essential to evaluate truly novel combinations, and organizations in these settings may be able to accurately assess such novelty. Similarly, while individual software developers may have difficulties in assessing the true novelty of an idea generated with GAI assistance, a larger group of developers in a complex software project may be better able to do so. Therefore, in these settings, it is conceivable that GAI tools may result in original products spanning more knowledge domains *and* being more novel, due to organizations' ability to accurately assess novelty.

In the *second* phase of innovation, GAI might be less effective in implementing ideas in physical innovation settings, due to the lack of a close integration of GAI outcome and (physical) products. Nonetheless, GAI tools may also assist idea implementation in industries like automotive or aviation, for instance by generating 3D-designs of physical products or components. Often, such 3D-designs serve as the starting point for production processes and can thus be an efficiency-enhancing element in the product implementation phase. Similarly, while GAI tools may not be able to directly create the actual products in non-digital settings, they can provide guidance and instruction on how to do this, which can be helpful for inexperienced innovators in less knowledge-intensive industries.

Hence, while this study provides valuable insights into the interplay between GAI and digital innovation, it also lays the groundwork for further explorations into how GAI may affect innovation in other fields and under varying conditions.

6.4 | Managerial and policy implications

Our findings offer practical implications for platform managers and policy makers. First, for platform governance, our results suggest that ecosystem managers like *Mozilla* may need to adapt their strategies to accommodate the surge in new add-ons enabled by GAI. Platforms will need more active curation or recommendation algorithms to ensure user discovery of genuinely new products. In particular, platforms such as *Firefox* may need to consider potential similarity with existing add-ons when deciding which add-ons are allowed on the platform and which ones are recommended to users. For instance, search ranking algorithms should be designed to penalize add-ons with limited novelty.

Second, our findings about the decline in updates to existing products highlight another important challenge for platform managers. As developers shift their attention from improvements to original products, users may be left with unmaintained products, which can deteriorate the overall user experience and, potentially, lead to security risks for users. Managers might consider incentives (e.g., ranking boosts or maintenance badges) for regularly updated add-ons, especially those that are important for the overall user experience and those that may be subject to security risks.



Third, there are also general implications for innovation policy that emerge from our findings. First, our finding that GAI assistance can lead to a higher number of new entrants points toward a democratization effect on (digital) innovation. From an innovation policy perspective, this may foster competition but also raise potential concerns about product quality and safety. Second, as newly introduced products may not necessarily be novel, policy makers may have to consider measures to ensure technological progress and avoid duplicate efforts that could lead to inefficiencies.

6.5 | Limitations and future work

The limitations of this study warrant further attention. First, as we discussed above, the study is confined to software product development, and future research could investigate the effects of GAI tools across different industries, which would provide a more comprehensive view of their impact on innovation. Second, the analysis only focuses on a relatively short time frame. This time frame may not capture the long-term impacts and evolving patterns of innovation following the introduction of GAI tools. Conducting a longitudinal study spanning several years could unearth the long-term effects on innovation. Third, our findings reflect the capabilities of current GAI models. Future (G)AI models may differ in their effect on innovation and may, for instance, be better able to generate truly novel ideas. Lastly, while our survey among *Firefox* developers is a valuable complement for our main analysis, it is important to acknowledge its relatively small sample size. Insights from this survey can shed light on potential mechanisms, but should primarily be interpreted in conjunction with our other analyses.

ACKNOWLEDGEMENT

Open access publication funding provided by COUPERIN CY26.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Erdem Doğukan Yılmaz  <https://orcid.org/0000-0002-2727-3557>

Tim Meyer  <https://orcid.org/0009-0003-3562-9327>

REFERENCES

- Acemoglu, D., & Restrepo, P. (2020). Robots and jobs: Evidence from US labor markets. *Journal of Political Economy*, 128(6), 2188–2244.
- Acemoglu, D., & Restrepo, P. (2024). *Automation and rent dissipation: Implications for wages, inequality, and productivity* (No. w32536). National Bureau of Economic Research.
- Agrawal, A., Gans, J. S., & Goldfarb, A. (2019). Exploring the impact of artificial intelligence: Prediction versus judgment. *Information Economics and Policy*, 47, 1–6.
- Agrawal, A., Gans, J. S., & Goldfarb, A. (2022). *Power and prediction: The disruptive economics of artificial intelligence*. Harvard Business Press.
- Agrawal, A., McHale, J., & Oettl, A. (2019). Finding needles in haystacks: Artificial intelligence and recombinant growth. In *The economics of artificial intelligence: An agenda* (pp. 149–174). University of Chicago.
- Agrawal, A., McHale, J., & Oettl, A. (2024). Artificial intelligence and scientific discovery: A model of prioritized search. *Research Policy*, 53(5), 104989.

- Ahuja, G., Lampert, C. M., & Novelli, E. (2013). The second face of appropriability: Generative appropriability and its determinants. *Academy of Management Review*, 38(2), 248–269.
- Ali, A., Kalwani, M. U., & Kovenock, D. (1993). Selecting product development projects: Pioneering versus incremental innovation strategies. *Management Science*, 39(3), 255–274.
- Allen, R., & Choudhury, P. (2022). Algorithm-augmented work and domain experience: The countervailing forces of ability and aversion. *Organization Science*, 33(1), 149–169.
- Allison, P. D., & Waterman, R. P. (2002). Fixed-effects negative binomial regression models. *Sociological Methodology*, 32(1), 247–265.
- Amabile, T. M. (1996). *Creativity and innovation in organizations* (Vol. 5). Harvard Business School.
- Angrist, J. D., & Pischke, J. S. (2009). *Mostly harmless econometrics: An Empiricist's companion*. Princeton University.
- Argote, L., Lee, S., & Park, J. (2021). Organizational learning processes and outcomes: Major findings and future research directions. *Management Science*, 67(9), 5399–5429.
- Arkhangelsky, D., Athey, S., Hirshberg, D. A., Imbens, G. W., & Wager, S. (2021). Synthetic difference-in-differences. *American Economic Review*, 111(12), 4088–4118.
- Arts, S., & Veugelers, R. (2015). Technology familiarity, recombinant novelty, and breakthrough invention. *Industrial and Corporate Change*, 24(6), 1215–1246.
- Banbury, C. M., & Mitchell, W. (1995). The effect of introducing important incremental innovations on market share and business survival. *Strategic Management Journal*, 16(S1), 161–182.
- Bennett, V. M., & Hall, T. A. (2020). Software availability and entry. *Strategic Management Journal*, 41(5), 950–962.
- Boudreau, K. J. (2012). Let a thousand flowers bloom? An early look at large numbers of software app developers and patterns of innovation. *Organization Science*, 23(5), 1409–1427.
- Boussiou, L., Lane, J. N., Zhang, M., Jacimovic, V., & Lakhani, K. R. (2024). The crowdless future? Generative AI and creative problem-solving. *Organization Science*, 35(5), 1589–1607.
- Bresnahan, T. F., & Trajtenberg, M. (1995). General-purpose technologies ‘engines of growth’? *Journal of Econometrics*, 65(1), 83–108.
- Brynjolfsson, E., Li, D., & Raymond, L. (2025). Generative AI at work. *The Quarterly Journal of Economics*, 140(2), 889–942.
- Brynjolfsson, E., Mitchell, T., & Rock, D. (2018). What can machines learn and what does it mean for occupations and the economy? *AEA Papers and Proceedings*, 108, 43–47.
- Brynjolfsson, E., Rock, D., & Syverson, C. (2021). The productivity J-curve: How intangibles complement general purpose technologies. *American Economic Journal: Macroeconomics*, 13(1), 333–372.
- Cameron, A. C., & Trivedi, P. K. (2013). *Regression analysis of count data* (Vol. 53). Cambridge University Press.
- Cennamo, C., & Santalo, J. (2013). Platform competition: Strategic trade-offs in platform markets. *Strategic Management Journal*, 34(11), 1331–1350.
- Chandy, R., Hopstaken, B., Narasimhan, O., & Prabhu, J. (2006). From invention to innovation: Conversion ability in product development. *Journal of Marketing Research*, 43(3), 494–508.
- Chandy, R. K., & Tellis, G. J. (1998). Organizing for radical product innovation: The overlooked role of willingness to cannibalize. *Journal of Marketing Research*, 35(4), 474–487.
- Chatterjee, J. (2017). Strategy, human capital investments, business-domain capabilities, and performance: A study in the global software services industry. *Strategic Management Journal*, 38(3), 588–608.
- Chatterji, A. K., & Fabrizio, K. R. (2014). Using users: When does external knowledge enhance corporate product innovation? *Strategic Management Journal*, 35(10), 1427–1445.
- Chen, J., & Roth, J. (2024). Logs with zeros? Some problems and solutions. *The Quarterly Journal of Economics*, 139(2), 891–936.
- Chen, L., Yi, J., Li, S., & Tong, T. W. (2022). Platform governance design in platform ecosystems: Implications for complementors' multihoming decision. *Journal of Management*, 48(3), 630–656.
- Chen, L., Zhang, P., Li, S., & Turner, S. F. (2022). Growing pains: The effect of generational product innovation on mobile games performance. *Strategic Management Journal*, 43(4), 792–821.
- Cockburn, I. M., Henderson, R., & Stern, S. (2018). The impact of artificial intelligence on innovation: An exploratory analysis. In *The economics of artificial intelligence: An agenda* (pp. 115–146). University of Chicago Press.



- Conti, R., Gambardella, A., & Mariani, M. (2014). Learning to be Edison: Inventors, organizations, and breakthrough inventions. *Organization Science*, 25(3), 833–849.
- Crafts, N. (2021). Artificial intelligence as a general-purpose technology: An historical perspective. *Oxford Review of Economic Policy*, 37(3), 521–536.
- Csaszar, F. A., & Levinthal, D. A. (2016). Mental representation and the discovery of new strategies. *Strategic Management Journal*, 37(10), 2031–2049.
- Dahlin, K. B., & Behrens, D. M. (2005). When is an invention really radical?: Defining and measuring technological radicalness. *Research Policy*, 34(5), 717–737.
- Dewar, R. D., & Dutton, J. E. (1986). The adoption of radical and incremental innovations: An empirical analysis. *Management Science*, 32(11), 1422–1433.
- Doshi, A. R., & Hauser, O. P. (2024). Generative AI enhances individual creativity but reduces the collective diversity of novel content. *Science Advances*, 10(28), eadn5290.
- Dosi, G. (1982). Technological paradigms and technological trajectories: A suggested interpretation of the determinants and directions of technical change. *Research Policy*, 11(3), 147–162.
- Eichenbaum, M. S., de Matos, M. G., Lima, F., Rebelo, S., & Trabandt, M. (2020). *Expectations, infections, and economic activity* (No. w27988). National Bureau of Economic Research.
- Eloundou, T., Manning, S., Mishkin, P., & Rock, D. (2024). GPTs are GPTs: Labor market impact potential of LLMs. *Science*, 384(6702), 1306–1308.
- Ethiraj, S. K., Kale, P., Krishnan, M. S., & Singh, J. V. (2005). Where do capabilities come from and how do they matter? A study in the software services industry. *Strategic Management Journal*, 26(1), 25–45.
- Fleming, L. (2001). Recombinant uncertainty in technological search. *Management Science*, 47(1), 117–132.
- Fleming, L., & Sorenson, O. (2004). Science as a map in technological search. *Strategic Management Journal*, 25(8–9), 909–928.
- Foerderer, J., Kude, T., Mithas, S., & Heinzl, A. (2018). Does platform owner's entry crowd out innovation? Evidence from Google photos. *Information Systems Research*, 29(2), 444–460.
- Furman, J. L., & Teodoridis, F. (2020). Automation, research technology, and researchers' trajectories: Evidence from computer science and electrical engineering. *Organization Science*, 31(2), 330–354.
- Gaessler, F., & Piezunka, H. (2023). Training with AI: Evidence from chess computers. *Strategic Management Journal*, 44(11), 2724–2750.
- Girotra, K., Meincke, L., Terwiesch, C., & Ulrich, K. T. (2023). Ideas are dimes a dozen: Large language models for idea generation in innovation. SSRN 4526071.
- Goldberg, S. G., Johnson, G. A., & Shriver, S. K. (2024). Regulating privacy online: An economic evaluation of the GDPR. *American Economic Journal: Economic Policy*, 16(1), 325–358.
- Goldfarb, A., Taska, B., & Teodoridis, F. (2023). Could machine learning be a general-purpose technology? A comparison of emerging technologies using data from online job postings. *Research Policy*, 52(1), 104653.
- Gourieroux, C., Monfort, A., & Trognon, A. (1984). Pseudo maximum likelihood methods: Applications to Poisson models. *Econometrica: Journal of the Econometric Society*, 52, 701–720.
- Greve, H. R. (2003). A behavioral theory of R&D expenditures and innovations: Evidence from shipbuilding. *Academy of Management Journal*, 46(6), 685–702.
- Griliches, Z. (1957). Hybrid corn: An exploration in the economics of technological change. *Econometrica*, 25(4), 501–522.
- Gruber, M., Harhoff, D., & Hoisl, K. (2013). Knowledge recombination across technological boundaries: Scientists vs. engineers. *Management Science*, 59(4), 837–851.
- He, V. F., Shrestha, Y. R., Puranam, P., & Miron-Spektor, E. (2024). *Searching together: A theory of human-AI co-creativity*. SSRN.
- Helfat, C. E., & Raubitschek, R. S. (2000). Product sequencing: Co-evolution of knowledge, capabilities and products. *Strategic Management Journal*, 21(10–11), 961–979.
- Helpman, E. (1998). *General-purpose technologies and economic growth*. MIT Press.
- Hoelzemann, J., Manso, G., Nagaraj, A., & Tranchero, M. (2024). *The streetlight effect in data-driven exploration* (No. w32401). National Bureau of Economic Research.
- Jeppesen, L. B., & Lakhani, K. R. (2010). Marginality and problem-solving effectiveness in broadcast search. *Organization Science*, 21(5), 1016–1033.

- Jia, N., Luo, X., Fang, Z., & Liao, C. (2024). When and how artificial intelligence augments employee creativity. *Academy of Management Journal*, 67(1), 5–32.
- Kaplan, S., & Vakili, K. (2015). The double-edged sword of recombination in breakthrough innovation. *Strategic Management Journal*, 36(10), 1435–1457.
- Katila, R., & Ahuja, G. (2002). Something old, something new: A longitudinal study of search behavior and new product introduction. *Academy of Management Journal*, 45(6), 1183–1194.
- Katz, M. L., & Shapiro, C. (1985). Network externalities, competition, and compatibility. *The American Economic Review*, 75(3), 424–440.
- Keijl, S., Gilsing, V. A., Knobben, J., & Duysters, G. (2016). The two faces of inventions: The relationship between recombination and impact in pharmaceutical biotechnology. *Research Policy*, 45(5), 1061–1074.
- Kijkuit, B., & van den Ende, J. (2007). The organizational life of an idea: Integrating social network, creativity and decision-making perspectives. *Journal of Management Studies*, 44(6), 863–882.
- Kim, S. (2025). *Navigating the rugged data landscape: The impact of data-extrapolation technologies on knowledge production*. Columbia Business School.
- Lawless, M. W., & Anderson, P. C. (1996). Generational technological change: Effects of innovation and local rivalry on performance. *Academy of Management Journal*, 39(5), 1185–1217.
- Levinthal, D. A., & March, J. G. (1993). The myopia of learning. *Strategic Management Journal*, 14(S2), 95–112.
- Lou, B., & Wu, L. (2021). AI on drugs: Can artificial intelligence accelerate drug development? Evidence from a large-scale examination of bio-pharma firms. *MIS Quarterly*, 45(3b), 1451–1482.
- MacCormack, A., Rusnak, J., & Baldwin, C. Y. (2006). Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, 52(7), 1015–1030.
- Malerba, F., & Orsenigo, L. (1996). Schumpeterian patterns of innovation are technology-specific. *Research Policy*, 25(3), 451–478.
- Mamoshina, P., Vieira, A., Putin, E., & Zhavoronkov, A. (2016). Applications of deep learning in biomedicine. *Molecular Pharmaceutics*, 13(5), 1445–1454.
- Martin, W., & Pham, C. S. (2020). Estimating the gravity model when zero trade flows are frequent and economically determined. *Applied Economics*, 52(26), 2766–2779.
- Miric, M., Boudreau, K. J., & Jeppesen, L. B. (2019). Protecting their digital assets: The use of formal & informal appropriability strategies by App developers. *Research Policy*, 48(8), 103738.
- Miric, M., & Jeppesen, L. B. (2020). Does piracy lead to product abandonment or stimulate new product development?: Evidence from mobile platform-based developer firms. *Strategic Management Journal*, 41(12), 2155–2184.
- Miric, M., Ozalp, H., & Yilmaz, E. D. (2023). Trade-offs to using standardized tools: Innovation enablers or creativity constraints? *Strategic Management Journal*, 44(4), 909–942.
- Nagle, F., & Teodoridis, F. (2020). Jack of all trades and master of knowledge: The role of diversification in new distant knowledge integration. *Strategic Management Journal*, 41(1), 55–85.
- Nerkar, A., & Roberts, P. W. (2004). Technological and product-market experience and the success of new product introductions in the pharmaceutical industry. *Strategic Management Journal*, 25(8–9), 779–799.
- Nohria, N., & Gulati, R. (1996). Is slack good or bad for innovation? *Academy of Management Journal*, 39(5), 1245–1264.
- Noy, S., & Zhang, W. (2023). Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654), 187–192.
- Ocasio, W. (1997). Towards an attention-based view of the firm. *Strategic Management Journal*, 18(S1), 187–206.
- Powell, D., & Seabury, S. (2018). Medical care spending and labor market outcomes: Evidence from workers' compensation reforms. *American Economic Review*, 108(10), 2995–3027.
- Raisch, S., & Krakowski, S. (2021). Artificial intelligence and management: The automation–augmentation paradox. *Academy of Management Review*, 46(1), 192–210.
- Rosenkopf, L., & Nerkar, A. (2001). Beyond local search: Boundary-spanning, exploration, and impact in the optical disk industry. *Strategic Management Journal*, 22(4), 287–306.
- Schumpeter, J. A. (1934). *The theory of economic development*. Harvard Economic Studies.
- Shane, S. (2000). Prior knowledge and the discovery of entrepreneurial opportunities. *Organization Science*, 11(4), 448–469.
- Silva, J. S., & Tenreyro, S. (2006). The log of gravity. *The Review of Economics and Statistics*, 88(4), 641–658.



- Silva, J. S., & Tenreiro, S. (2011). Further simulation evidence on the performance of the Poisson pseudo-maximum likelihood estimator. *Economics Letters*, 112(2), 220–222.
- Song, P., Xue, L., Rai, A., & Zhang, C. (2018). The ecosystem of software platform: A study of asymmetric cross-side network effects and platform governance. *MIS Quarterly*, 42(1), 121–142.
- Teodoridis, F., Bikard, M., & Vakili, K. (2019). Creativity at the knowledge frontier: The impact of specialization in fast-and slow-paced domains. *Administrative Science Quarterly*, 64(4), 894–927.
- Tiwana, A. (2008). Does technological modularity substitute for control? A study of alliance performance in software outsourcing. *Strategic Management Journal*, 29(7), 769–780.
- Tiwana, A. (2015). Evolutionary competition in platform ecosystems. *Information Systems Research*, 26(2), 266–281.
- Tranchoero, M. (2024). Finding diamonds in the rough: Data-driven opportunities and pharmaceutical innovation. Working Paper. https://www.matteotranchoero.com/pdf/Matteo_Tranchoero_JMP_latest.pdf
- Turner, S. F., Mitchell, W., & Bettis, R. A. (2013). Strategic momentum: How experience shapes temporal consistency of ongoing innovation. *Journal of Management*, 39(7), 1855–1890.
- van der Geest, C., & van Angeren, J. (2023). Architectural generativity: Leveraging complementor contributions to the platform architecture. *California Management Review*, 65(2), 71–92.
- Von Krogh, G. (2018). Artificial intelligence in organizations: New opportunities for phenomenon-based theorizing. *Academy of Management Discoveries*, 4(4), 404–409.
- Weisberg, R. W. (1999). Creativity and knowledge: A challenge of theories. In *Handbook of creativity* (pp. 226–250). Cambridge University Press.
- Wen, W., & Zhu, F. (2019). Threat of platform-owner entry and complementor responses: Evidence from the mobile app market. *Strategic Management Journal*, 40(9), 1336–1377.
- Wu, B., Wan, Z., & Levinthal, D. A. (2014). Complementary assets as pipes and prisms: Innovation incentives and trajectory choices. *Strategic Management Journal*, 35(9), 1257–1278.
- Wu, L., Hitt, L., & Lou, B. (2020). Data analytics, innovation, and firm productivity. *Management Science*, 66(5), 2017–2039.
- Xiao, T., Makhija, M., & Karim, S. (2022). A knowledge recombination perspective of innovation: Review and new research directions. *Journal of Management*, 48(6), 1724–1777.
- Yilmaz, E. D., Naumovska, I., & Aggarwal, V. A. (2023). *AI-driven labor substitution: Evidence from Google translate and ChatGPT*. SSRN.
- Yilmaz, E. D., Naumovska, I., & Miric, M. (2023). Does imitation increase or decrease demand for an original product? Understanding the opposing effects of discovery and substitution. *Strategic Management Journal*, 44(3), 639–671.
- Zahavi, T., & Lavie, D. (2013). Intra-industry diversification and firm performance. *Strategic Management Journal*, 34(8), 978–998.
- Zahavy, T. (2026). LLMs can't jump. PhilSci-Archive. <https://philsci-archive.pitt.edu/28024/>

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Yilmaz, E. D., & Meyer, T. (2026). The impact of generative artificial intelligence on innovation: Evidence from software products. *Strategic Management Journal*, 1–50. <https://doi.org/10.1002/smj.70118>

APPENDIX A: ADDITIONAL FIGURES

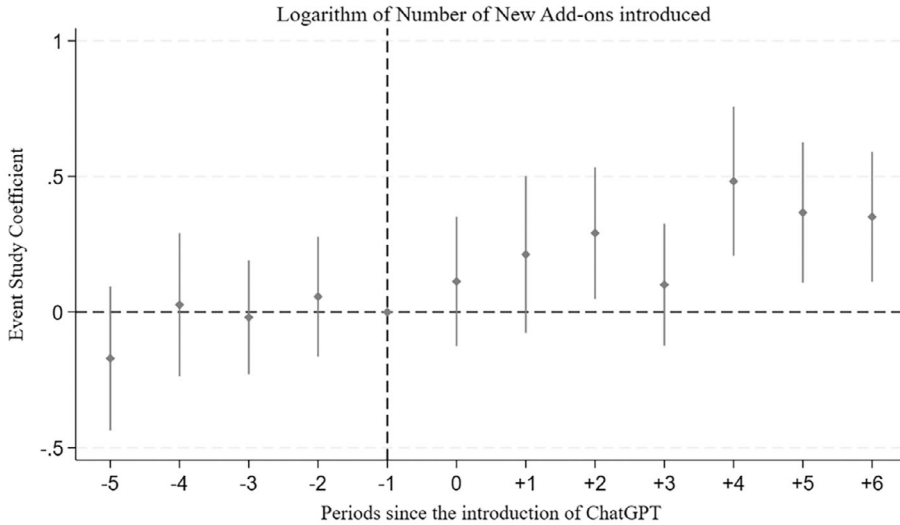


FIGURE A1 Logarithm of the number of new add-ons introduced: leads and lags of the treatment. This figure plots estimates (and confidence intervals) of β_1 from the event study specification. Periods since treatment are represented on the horizontal axis, and the outcome variable on the vertical axis is the logarithm of the number of new add-ons introduced. Coefficients (and 95 percent confidence intervals) from log-OLS models are presented.

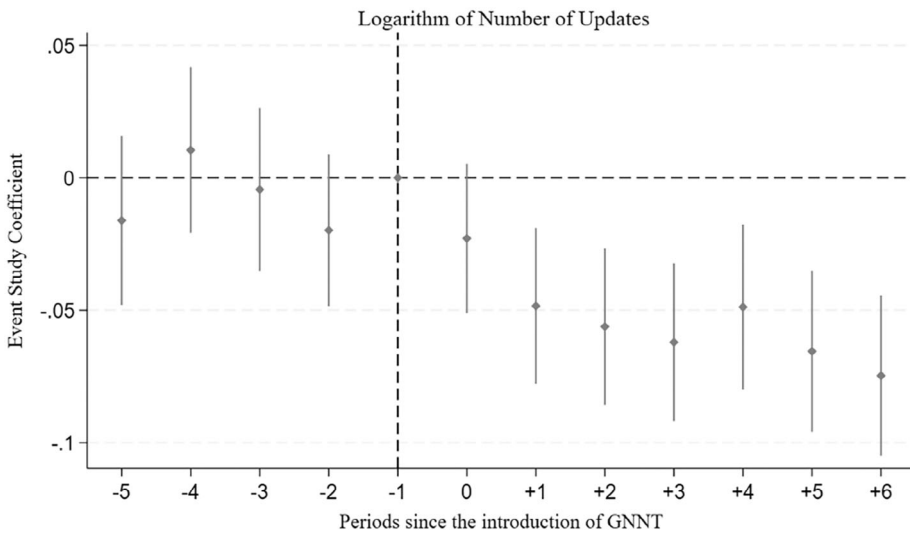


FIGURE A2 Logarithm of the number of updates: leads and lags of the treatment. This figure plots estimates (and confidence intervals) of β_1 from the event study specification. Periods since treatment are represented on the horizontal axis and the outcome variable on the vertical axis is the logarithm of the number of add-on updates introduced. Coefficients (and 95 percent confidence intervals) from log-OLS models are presented.



APPENDIX B: EFFECT HETEROGENEITY

B.1 | Minor versus Major Updates

If the availability of GAI assistance does indeed lead developers to pursue more substantial innovation, we should also observe that not all updates are affected equally. Specifically, developers may refrain from more minor updates to their add-ons but still pursue more major updates that involve more significant changes. In Table B1, we therefore focus on the impact of *ChatGPT* on the number of updates, distinguishing between major and minor updates.²⁷ Major updates usually represent significant improvements in a software application's quality or user experience, marking a shift from, say, version 1.0 to version 2.0. These updates often introduce new features or improved functionalities. Conversely, minor updates address more trivial issues such as bug fixes or resolve incompatibility problems, and are usually represented as a transition from, say, version 1.0.0 to version 1.0.1.

We again employ the OLS (1,5), Log-OLS (2,6), Poisson (3,7), and Logit (4,8) models to compare the number of major and minor updates, the logarithm of the number of updates, and whether a (major or minor) update occurred (0/1) following the treatment. In the case of major updates, the *Post* $\times$ *Treatment* coefficients in columns (1) to (4) show no effect (e.g., column (1): $\beta = -0.0012$, $p = .9607$). Minor updates, on the other hand, show a different pattern. The *Post* $\times$ *Treatment* coefficients in columns (5) to (8) provide evidence for a negative effect (e.g., column (5): $\beta = -0.1067$, $p < .0001$), implying a substantial decrease in the number of minor updates following the introduction of *ChatGPT*. These results suggest that, while the introduction of *ChatGPT* resulted in a reduction in the number of minor updates introduced, there is no evidence for a change in the number of major updates introduced.

B.2 | Competitive Value

Our reasoning behind Hypothesis 3 rests on the idea that developers allocate their efforts to original or generational innovation depending on the associated incentives, which in turn are related to the opportunity to engage in original innovation and to considerations related to competitive threats. We would therefore expect that the extent to which developers shift their efforts from software improvements to original software products changes as these incentives change. Specifically, if improvements to existing software products are valuable in fighting competition and preserving competitive advantage, we would expect developers to be less likely to shift their efforts away from them. For instance, in our context, the size of the existing user base likely plays a crucial role in the trade-off between focusing on original software products and software improvements. Products with a large user base create strong commercial incentives to continue improving these products, for instance due to established network effects and high switching costs for users (Cennamo & Santalo, 2013; Katz & Shapiro, 1985). In these cases, the risk of disrupting existing users and giving up existing investments such as professional development teams, may outweigh the benefits of developing original products, even with GAI assistance, and may push developers to keep engaging in software improvements. Therefore, we would expect that the negative relationship between GAI availability and the number of improvements to existing software products will be moderated by the competitive value of the software product, such that the relationship will be weaker for products with greater competitive value.

²⁷Summary statistics for these analyses are provided in Supporting Information S1, Tables S1.5 and S1.6.

TABLE B1 Heterogeneous Effect: Major versus Minor Updates.

	(1) Number of major updates	(2) Logarithm of number of major updates	(3) Number of major updates	(4) Major update (0/1)	(5) Number of minor updates	(6) Logarithm of number of minor updates	(7) Number of minor updates	(8) Minor update (0/1)
Post × Treatment	−0.0012 (0.0251)	−0.0014 (0.0157)	−0.0190 (0.1923)	−0.0187 (0.2035)	−0.1067 (0.0230)	−0.0561 (0.0105)	−0.2791 (0.0574)	−0.3602 (0.0549)
Constant	[0.9607]	[0.9301]	[0.9213]	[0.9266]	[0.0000]	[0.0000]	[0.0000]	[0.0000]
	0.1308	0.0866	−1.7338		0.4263	0.2360	−0.3989	
	(0.0073)	(0.0046)	(0.0614)		(0.0067)	(0.0031)	(0.0141)	
	[0.0000]	[0.0000]	[0.0000]		[0.0000]	[0.0000]	[0.0000]	
Add-on- treatment FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Adjusted R ²	0.052	0.020			0.174	0.173		
Pseudo R ²			0.095	0.012			0.200	0.003
Log-likelihood	−1535.0	221.9	−1581.7	−977.4	−40015.2	−13790.5	−30762.9	−12192.8
Observations	3,888	3,888	3,888	3,888	33,792	33,792	33,792	33,684

Note: Unit of observation: *Add-on(i)*. Models: *OLS (1,5)*; *Log-OLS (2,6)*; *Poisson(3,7)*; *Logit (4,8)*. Standard errors are robust to heteroskedasticity, clustered at the add-on-treatment group pair level, and reported in parentheses. *p*-Values are reported in brackets. The sample sizes differ because add-ons are required to have at least one major update or minor update in both the treatment and control groups to avoid collinearity with add-on-treatment fixed effects.



TABLE B2 Heterogeneous effect: competitive value of add-ons.

	(1) Number of updates	(2) Logarithm of number of updates	(3) Number of updates	(4) Updated (0/1)	(5) Number of updates	(6) Logarithm of number of updates	(7) Number of updates	(8) Updated (0/1)
Post × Treatment	−0.0964 (0.0251) [0.0001]	−0.0462 (0.0110) [0.0000]	−0.2263 (0.0556) [0.0000]	−0.285 (0.0558) [0.0000]	−0.0999 (0.0252) [0.0001]	−0.0480 (0.0110) [0.0000]	−0.236 (0.0559) [0.0000]	−0.295 (0.0561) [0.0000]
Post × Treatment × User base size					9.71e−08 (0.0000) [0.0000]	4.87e−08 (0.0000) [0.0000]	1.73e−07 (0.0000) [0.0000]	2.34e−07 (0.0000) [0.0000]
Constant	0.4704 (0.0073) [0.0000]	0.2551 (0.0032) [0.0000]	−0.2703 (0.0139) [0.0000]		0.470 (0.0073) [0.0000]	0.255 (0.0032) [0.0000]	−0.271 (0.0139) [0.0000]	[0.0713]
Add-on-Treatment FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Adjusted R^2	0.201	0.213			0.202	0.213		
Pseudo R^2			0.220	0.002			0.220	0.002
Log-likelihood	−40779.1	−14077.0	−32033.4	−11806.0	−40779.1	−14077.0	−32033.4	−11806.0
Observations	32,616	32,616	32,616	32,436	32,616	32,616	32,616	32,436

Note: Unit of observation: *Add-on(i)*. Models: *OLS (1,4)*; *Log-OLS (2,5)*; *Poisson(3,6)*; *Logit (4,8)*. Standard errors are robust to heteroskedasticity, clustered at the add-on-treatment group pair level and reported in parentheses. *p*-Values are reported in brackets. The sample is smaller as the variable *User Base Size* is not available for all add-ons one month before the treatment. Models 1 to 4 replicate the results in Table 5 with a smaller sample. Models 5 to 8 show how the effect changes depending on the size of the user base.

To test this idea, we replicate our main analysis (i.e., Table 5) and add the competitive value of add-ons as a moderator. Specifically, we draw on the *User Base Size*, i.e., the number of active users of a given add-on in October 2022, a month before the treatment.²⁸ Table B2 presents the heterogeneous effects depending on add-on *User Base Size*. We again use four models—OLS (1,4), Log-OLS (2,5), Poisson (3,6), and Logit (4,8)—to compare the number of updates, the logarithm of the number of updates, and whether an update occurred (0/1) following the treatment. The first four models replicate the results of Table 5 with a smaller sample.²⁹ The *Post* × *Treatment* coefficients in columns (1) to (4) show a negative effect (e.g., column (1): $\beta = -0.0964$, $p = .0001$), consistent with the analyses reported in Table 5. Columns (5) to (8) show how the effect changes depending on the *User Base Size*. Notably, the interaction term (*Post* × *Treatment* × *User Base Size*) is positive (e.g., column (5): $\beta = 9.71\text{e-}08$, $p = .0074$) in all four models, implying that the decrease in the number of updates is indeed less pronounced for add-ons with larger user bases.³⁰ Taken together, these findings are consistent with the idea that the (re)allocation of effort between original and generational innovation is driven, at least in part, by developer incentives.

It is important to note that user-base size can be correlated with professional development teams that may have stronger motivations to update their add-ons to maintain profitability. For this reason, in Supporting Information S9, we replicate our analysis with a smaller sample that excludes professional developer teams and find that our results hold. Hence, our results are consistent with the idea that large user bases create stronger retention incentives, whether or not the developer is a professional development team.

APPENDIX C: EXCLUDING ADD-ONS WITH GAI FUNCTIONALITY

It is possible that the effects we observe are not only driven by developers using GAI for programming (i.e., by supply side factors), but also by an increased demand for add-ons with GAI functionalities (i.e., by demand side factors). To investigate this possibility, we first compiled a dictionary of AI-related keywords and proceeded to search for these keywords within the descriptions of various add-ons. The keywords employed in this search include terminologies central to the fields of artificial intelligence and machine learning. These are:

- Artificial Intelligence: A broad term encompassing techniques that enable machines to mimic human cognitive functions.

²⁸Summary statistics for these analyses are provided in Supporting Information S1, Table S1.3. We measure User Base Size in October 2022, one month before the treatment, as the size itself may be influenced by the treatment, which could lead to a 'Bad Control' issue, resulting in biased coefficient estimates, as pointed out by Angrist and Pischke (2009).

²⁹The sample size for this analysis is smaller as the user base size was not available for all add-ons.

³⁰For the effect to be positive (i.e., for the number of updates to increase post-treatment), the size of the user base would need to be large enough to counterbalance the negative *Post* × *Treatment* coefficient. To calculate the user base size that would lead to a positive effect, we can set the sum of the *Post* × *Treatment* coefficient and the product of the *Post* × *Treatment* × *User Base Size* coefficient and the user base size to zero. In column (5), for example, this gives: $-0.0999 + (9.71\text{e-}08 \times \text{User Base Size}) = 0$. Solving for *User Base Size*, we get: $\text{User Base Size} = 0.0999/9.71\text{e-}08 \approx 1,028,836$. Therefore, according to column (5), the user base size would need to be 1,028,836 for the number of updates to increase following the introduction of *ChatGPT*. In our sample, there are only 10 add-ons for which the introduction of *ChatGPT* would have a positive effect.



TABLE C1 Results for number of new add-on introductions when add-ons mentioning AI are removed from the sample.

	(1) New add-ons introduced	(2) Log. new add-ons introduced	(3) New add-ons introduced
Post × Treatment	6.0971 (1.8778) [0.0029]	0.2599 (0.0712) [0.0010]	0.2093 (0.0523) [0.0001]
Constant	24.1022 (0.5477) [0.0000]	2.9703 (0.0208) [0.0000]	3.3673 (0.0187) [0.0000]
Category-treatment FE	Yes	Yes	Yes
Month FE	Yes	Yes	Yes
Adjusted R^2	0.884	0.829	
Pseudo R^2			0.621
Log-likelihood	−1156.6	−18.35	−2827.5
Observations	360	360	360

Note: Unit of observation: *Category(i)*. Models: *OLS*; *Log-OLS*; *Poisson*. Standard errors are robust to heteroskedasticity, clustered at the category-treatment group pair level and reported in parentheses. *p*-Values are reported in brackets.

- *TensorFlow*: An open-source machine learning framework developed by *Google*, primarily used for deep learning.
- *ChatGPT*, *Bing Chat*, *Bard*: Names of various AI models and tools related to natural language processing.
- Machine Learning: A subset of AI that includes statistical techniques to enable machines to improve tasks with experience.
- Deep Learning: A subset of machine learning that involves algorithms inspired by the structure and function of the brain.
- Neural Network: A series of algorithms that attempts to recognize underlying relationships in a dataset through processes that mimic human brain operations.
- Natural Language Processing: A field focusing on the interaction between computers and humans through natural language.
- LLM: An abbreviation for Large Language Models, a foundational technology for many GAI applications.
- NLP: An abbreviation for Natural Language Processing.
- Computer Vision: The field that trains machines to interpret and make decisions based on visual data.
- Reinforcement Learning: A type of machine learning algorithm that is based on the idea of reward feedback.
- GAN: Short for Generative Adversarial Networks, a class of machine learning models designed to generate new data instances.

TABLE C2 Results for number of new add-on introductions when add-ons identified by *ChatGPT* Chat Completion as being AI-related are removed from the sample.

	(1) New add-ons introduced	(2) Log. new add-ons introduced	(3) New add-ons introduced
Post × Treatment	6.1778 (2.0283) [0.0049]	0.2500 (0.0607) [0.0003]	0.2089 (0.0536) [0.0001]
Constant	24.1556 (0.5071) [0.0000]	2.9735 (0.0152) [0.0000]	3.3730 (0.0167) [0.0000]
Category-treatment FE	Yes	Yes	Yes
Month FE	Yes	Yes	Yes
Adjusted R^2	0.887	0.833	
Pseudo R^2			0.625
Log-likelihood	−1154.0	−16.97	−2840.6
Observations	360	360	360

Note: Unit of observation: *Category(i)*. Models: *OLS*; *Log-OLS*; *Poisson*. Standard errors are robust to heteroskedasticity, clustered at category-treatment group pair level and reported in parentheses. *p*-Values are reported in brackets.

A *Python* script iteratively searched through the descriptions of add-ons for occurrences of the predefined keywords, counted these occurrences and saved them.

This effort led us to identify AI-related add-ons, i.e., add-ons that feature or promote AI functionalities. Out of the 3471 newly introduced add-ons in the post-treatment period, 144 were found to be AI-related (4.1%). Interestingly, this constitutes a 7-fold increase in the presence of AI-related add-ons. It is noteworthy that from the initial launch of the first add-on in 2004 until October 2022 (a month before *ChatGPT*'s launch), only 20 AI-related add-ons had entered the market.

We then exclude these AI-related add-ons from the sample and conduct a replication of the main analysis. The results remained consistent after this adjustment, as reported in Table C1.³¹

Given the importance of this issue, we also conduct an additional analysis to further ensure the robustness of our findings. Specifically, we use the *ChatGPT* Chat Completion model API, which is an AI system designed to process a written prompt and generate a detailed, contextually relevant response in natural language. In our case, we provided it with the descriptions of all add-ons in our dataset and asked it to classify each of them as “AI-related” or “Non-AI-related”. This model can integrate the meaning of the entire description, not just match keywords, and therefore can identify AI-related tools even when the link to AI is indirect or expressed in less obvious terms. We then removed all AI-related tools from our sample and replicate our main analysis. The results remain consistent with the main analysis, meaning that we again find evidence for an increase in the supply of new add-ons (see Table C2).

³¹Summary statistics for these analyses are provided in Supporting Information S1, Table S1.7.



APPENDIX D: DEVELOPER SURVEY AND MECHANISM CHECKS

To complement our quantitative analyses and gain deeper insights into the mechanisms driving our findings, we designed and implemented an online survey of *Firefox* add-on developers. The primary objective of the survey was to directly capture developers' experiences with generative AI tools and to triangulate the patterns observed in our main empirical analysis. For sample selection and data collection, we identified potential participants through the *Upwork* platform. To ensure that our sample consisted of developers with relevant pre-existing experience, we established a key selection criterion: each participant must have published at least one *Firefox* add-on prior to the widespread availability of GAI tools.

The survey instrument was structured into four thematic sections designed to provide a comprehensive view of the developers' context and their interaction with GAI. The initial section gathered data on developer background and experience, including their tenure in add-on development, professional status, and primary programming languages. The second section focused on generative AI usage and adoption, probing the timing of adoption, frequency

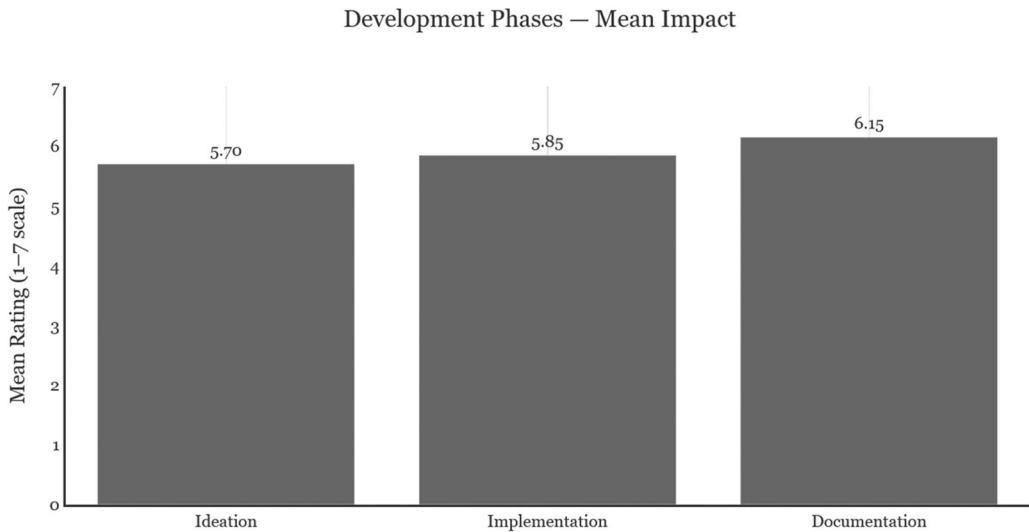


FIGURE D1 Perceived impact across development phases.

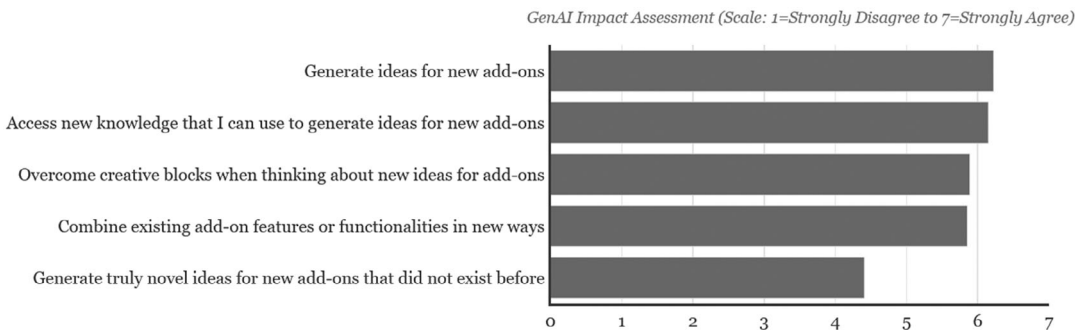


FIGURE D2 Perceived impact in the ideation phase.

of use, and the specific development tasks for which these tools are employed. The third section investigated the impact on development activities, specifically exploring how GAI tools have influenced developers' allocation of effort between new product creation and the maintenance of existing products. Finally, the survey explored innovation outcomes and perceptions, examining how developers perceive GAI's influence on their capacity for innovation and their resulting competitive strategies.

D.1. | Survey participants

We received initial responses from 37 developers. To ensure the reliability of our data, the survey instrument included an attention-check question that required respondents to select a specific answer. After filtering out 4 respondents who failed this check, our final sample consists of 33 developers. This process helps to make sure that the responses analyzed are from participants who were attentive and engaged throughout the survey.

The respondents represent a mix of experience levels, with a majority having between 1 and 6 years of software development experience (66.6%). The sample is composed predominantly of professional developers (84.8%), many of whom work as solo developers (37.5%) or in very small teams, underscoring the relevance of our findings for individual and small-scale innovation. The adoption of GAI among this group was both rapid and deep; a majority of respondents began using these tools within a few weeks of their availability (68.8%), and a significant majority now use them on a daily or near-daily basis in their workflow (84.9%).

D.2. | Perceived impact across development phases

When asked to rate the helpfulness of GAI across the entire development lifecycle (on a Likert Scale from 1 to 7), the results suggest that developers perceive GAI to be useful in all phases.

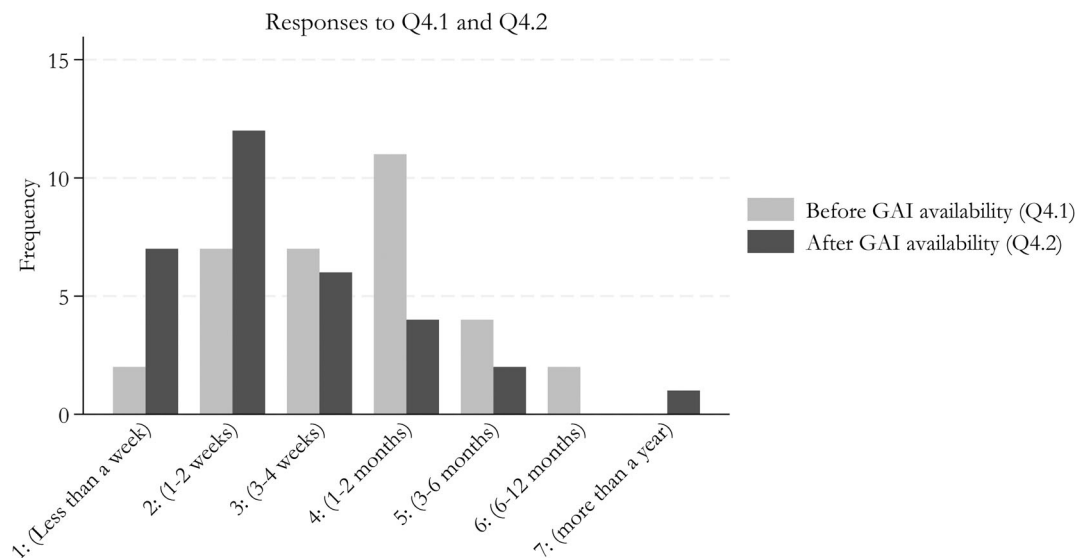


FIGURE D3 Estimated time required to develop browser add-ons (histogram).



While GAI is seen as helpful in the ideation phase ($M = 5.70$), its perceived utility is even slightly higher for implementation ($M = 5.85$) and for the documentation phase ($M = 6.15$), which includes tasks like writing release notes and summarizing changes (Figure D1).

D.3. | Ideation and efficient recombination mechanism

The survey provides robust support for GAI's role in the ideation phase (Figure D2). An overwhelming majority of developers reported using GAI for ideation (81.8%). Respondents strongly agreed that GAI helps them generate ideas for new add-ons ($M = 6.22$, $SD = 1.19$), overcome creative blocks ($M = 5.89$, $SD = 1.22$), and access new knowledge ($M = 6.15$, $SD = 1.20$). Crucially, this aligns with our concept of efficient recombination; developers agreed that GAI helps them combine existing features in new ways ($M = 5.85$, $SD = 1.10$). However, they were significantly more neutral about GAI's ability to help them generate *truly novel* ideas that did not exist before ($M = 4.41$, $SD = 1.45$). As one developer explained, this creates a distinction between personal and market novelty: *“The tools I created through brainstorming with Claude etc are usually novel to me. However, I later discover that some tools with similar functionalities already exist in the market.”* This key distinction supports our central argument: GAI is good at making the recombination of known components more efficient, thus increasing the *quantity* of original ideas, without necessarily increasing their fundamental *novelty*.

D.4. | Implementation mechanism

The survey results indicate that GAI's impact on the implementation phase is even more pronounced, with 97% of developers using it for this purpose. Respondents reported that GAI helps

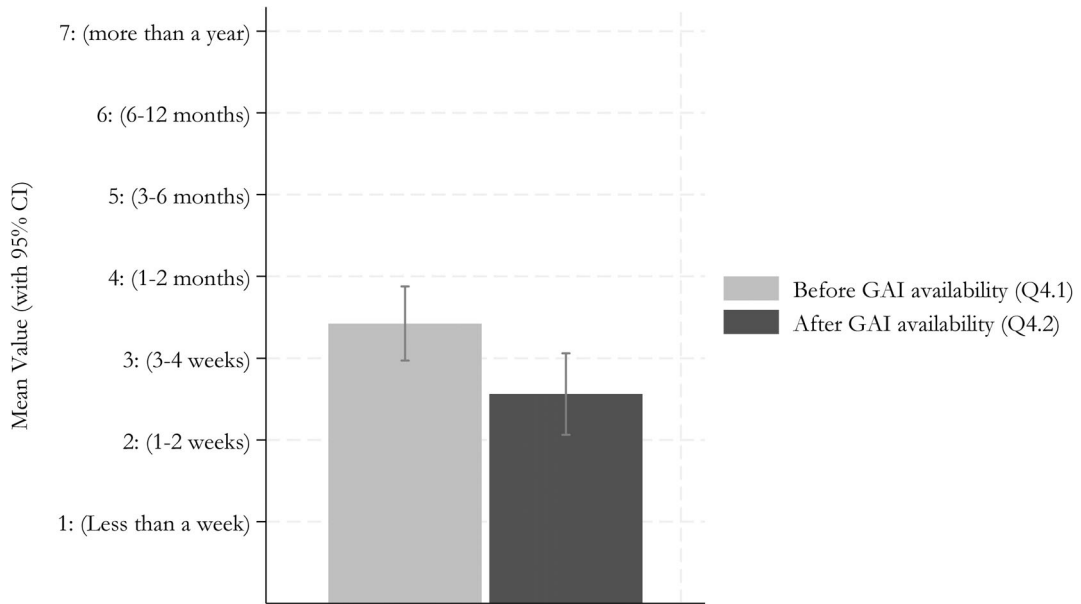


FIGURE D4 Estimated time required to develop browser add-ons (mean comparison).

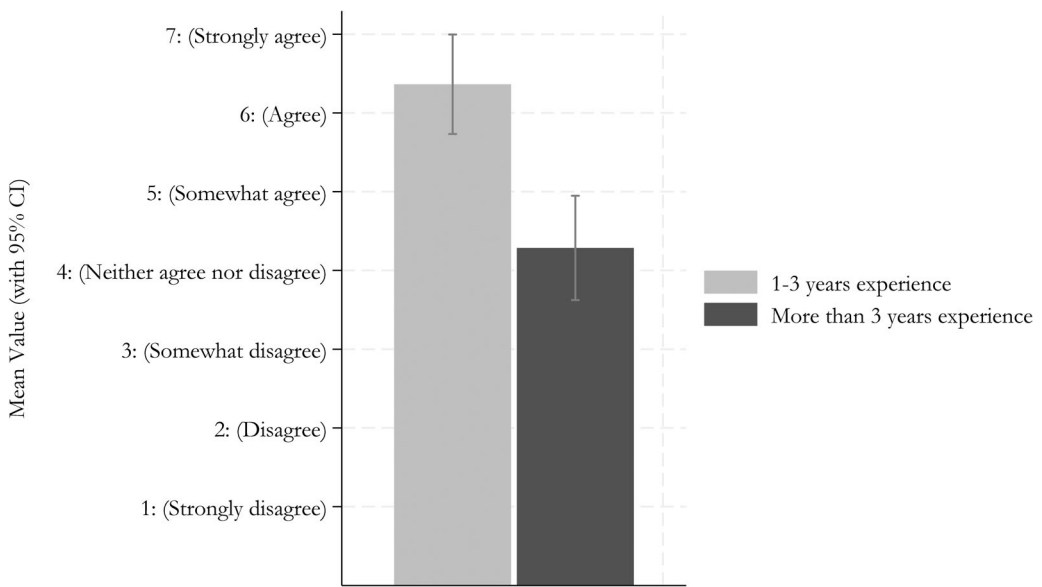


FIGURE D5 "GenAI helps me program add-on functionalities that I previously lacked the skills to develop." (mean comparison).

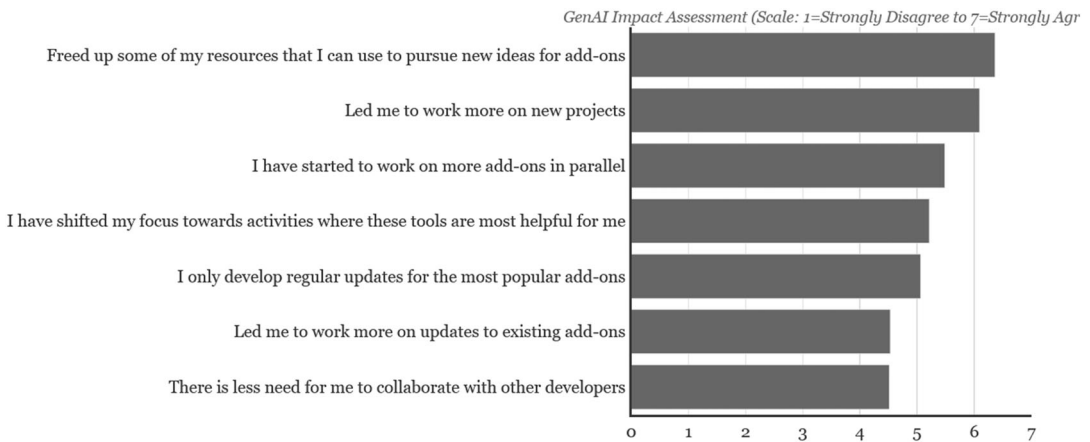


FIGURE D6 Reallocation of effort.

them write code ($M = 6.22$, $SD = 1.01$) and, most notably, write code *faster* ($M = 6.28$, $SD = 1.30$). This efficiency gain is a cornerstone of our argument, as it directly lowers the cost and time required to turn an idea into a product. This is further evidenced by a dramatic self-reported shift in development timelines: before GAI, only 27.3% of developers completed their most recent project in under 2 weeks, whereas after GAI, this figure jumped to 59.4%. The figure below visually represents this shift, showing the distribution of development times before and after GAI availability. The average reported development duration before GAI availability was 3.47 on a 7-point scale, which decreased to 2.56 after GAI became available (Figures D3 and D4).

Furthermore, GAI appears to lower skill-based barriers to entry, as developers agreed that it helps them program functionalities they previously lacked the skills to develop ($M = 5.00$, $SD = 1.67$). This is illustrated in the figure below, which shows the responses to the statement: “GenAI helps me program add-on functionalities that I previously lacked the skills to develop” The impact of GenAI tools varies with a developer's experience. Less experienced developers agree with this statement much more than experienced developers ($M(1-3 \text{ years experience}) = 6.36$; $M(\text{more than 3 years experience}) = 4.29$; $t = 4.13$; $p = .0001$) (Figure D5).

Related to this, one developer noted, “It has lowered the barrier to trying out new ideas and has made solo development more manageable and scalable.”

Less experienced developers also agree more strongly with the statement that “The availability of GenAI tools in add-on development has made it easier to enter the add-on market” than experienced developers (mean for 1–3 years experience = 6.45; mean for more than 3 years experience = 5.68; $t = 1.38$; $p = .0891$). This is further corroborated by some of the developers' statements. One developer mentioned that “I learnt everything related to programming from ChatGPT. The most frequently used prompt for me was please explain this as if you are explaining to a dummy.” While another one stated that “I think if you have truly novel ideas but lack the skills or time, GenAI can really help you bring your projects to life.” Together, these results are in line with our theory that GAI assistance can increase the number of new developers by lowering entry barriers.

D.5. | Reallocation of effort and competitive dynamics

The efficiencies gained in ideation and implementation directly translate into a strategic reallocation of developer effort (Figure D6). Developers strongly agreed that the availability of GAI has led them to work more on *new* add-on projects ($M = 6.09$, $SD = 1.15$) and that it has freed up resources to pursue new ideas ($M = 6.36$, $SD = 0.96$). In stark contrast, they were

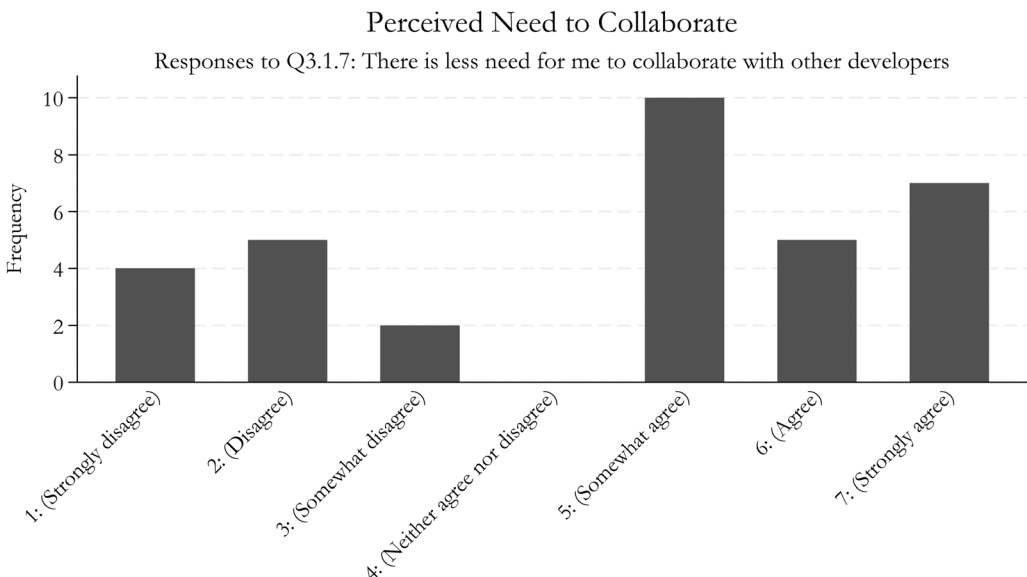


FIGURE D7 Impact on collaboration.

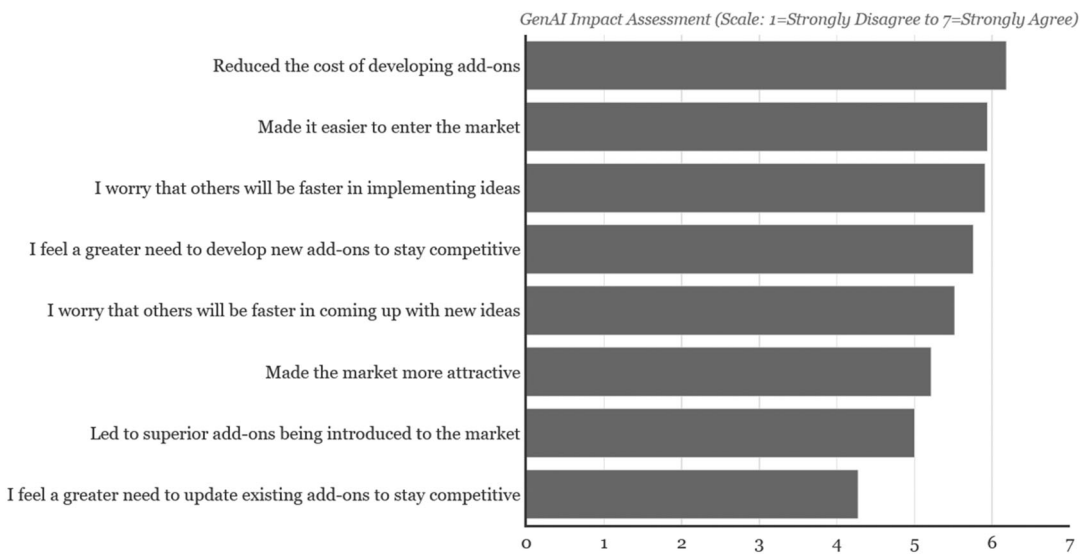


FIGURE D8 Market environment.

ambivalent about whether GAI led them to work more on *updates* to existing add-ons ($M = 4.53$, $SD = 1.81$). One developer powerfully summarized the resulting economic logic: “*I think AI dropped the sunk cost almost to zero. So, you can launch 10 apps at a time and see which ones fly and then focus more on those.*”

This shift is further explained by perceived market dynamics (see Figure D8). Developers feel a greater need to develop *new* add-ons to stay competitive ($M = 5.76$, $SD = 1.15$), driven by a worry that others will be faster in both conceiving ($M = 5.52$, $SD = 1.70$) and implementing ($M = 5.91$, $SD = 1.33$) new ideas. Conversely, they reported a significantly weaker competitive pressure to *update* existing add-ons ($M = 4.27$, $SD = 1.51$). This confirms our hypothesis that the opportunity to create original products with GAI, coupled with increased competitive threats in that domain, shifts attention and resources away from generational innovation.

D.6. | Impact on collaboration

The survey also explored whether GAI reduced the need for collaboration among developers. The results were mixed, suggesting this is not a primary driver of the observed changes in innovation patterns. The figure below shows the distribution of responses to the statement: “*Since GenAI tools have become available, there is less need for me to collaborate with other developers.*” (Figure D7).

D.7. | Market environment

The data reveals that developers perceive GAI as a powerful tool for lowering the barriers to innovation (Figure D8). Respondents strongly agreed that GAI has reduced the cost of developing add-ons and made it easier to enter the market. This reduction in cost and effort has

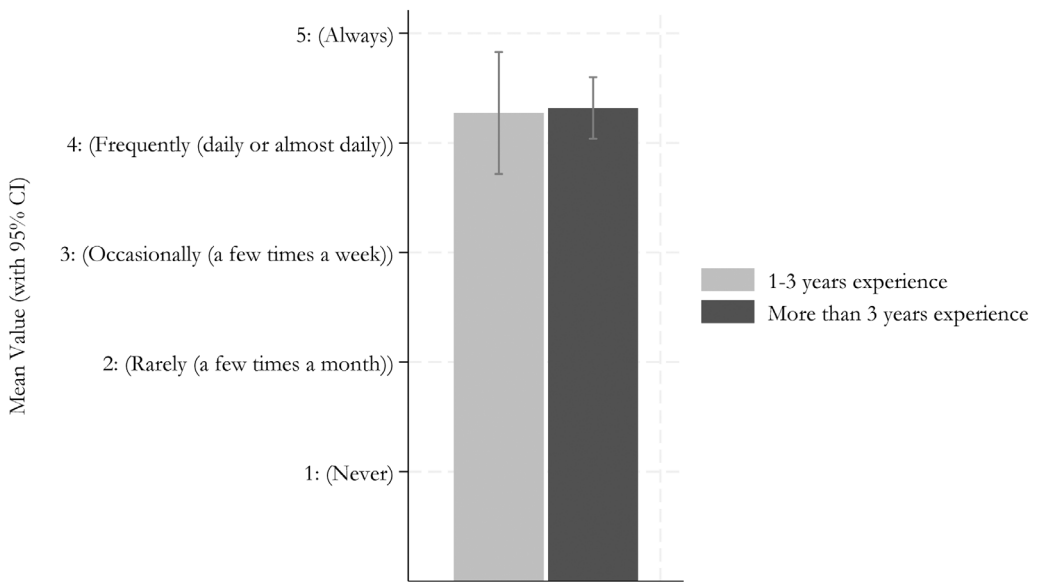


FIGURE D9 Generative AI usage of new versus existing developers. Response to Q2.1: How frequently do you use Generative AI tools (e.g., ChatGPT, Bard) in your software development workflow?

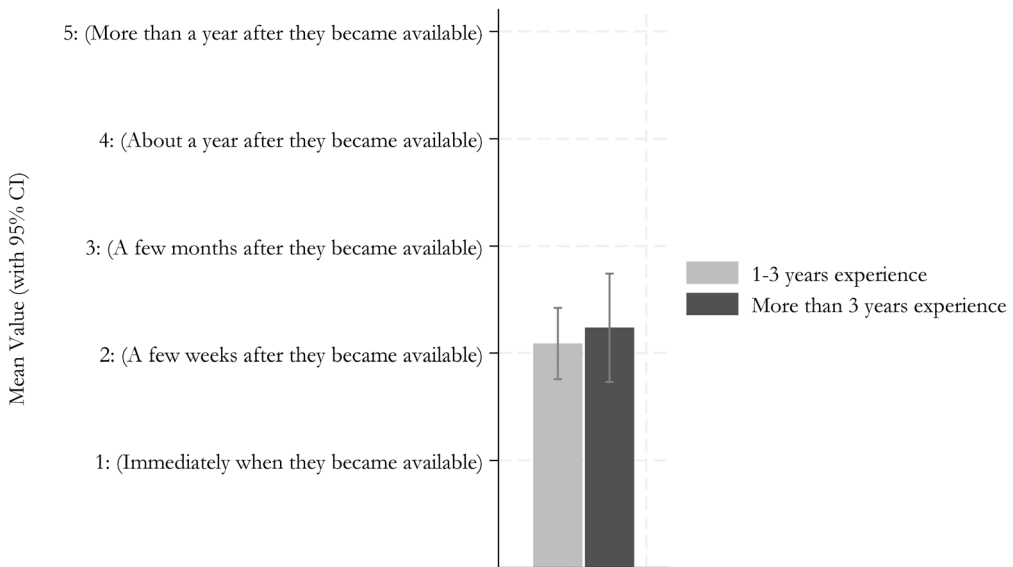


FIGURE D10 Adoption time of new versus existing developers. Response to Q2.2: When have you started using Generative AI tools (e.g., ChatGPT, Bard) in your software development workflow?

intensified competition around new creations. Developers reported feeling a greater need to develop new add-ons to stay competitive (Mean Rating = 5.76). This feeling is fueled by a worry that competitors using GAI will be faster at both implementing new ideas ($M = 5.91$) and conceiving them ($M = 5.52$).

Based on the survey of *Firefox* add-on developers, the chart below illustrates how Generative AI (GAI) has reshaped their strategic priorities and the competitive landscape. The findings highlight a significant shift in focus toward creating new products, driven by economic and competitive pressures.

In stark contrast to the pressure to create new products, developers reported a significantly weaker competitive need to update their existing add-ons ($M = 4.27$). This is the lowest-rated item on the chart, indicating that the strategic focus has moved away from incremental improvements and toward original product launches.

D.8. | Potential differences in adoption between new and existing developers

As it is possible that new developers use GAI more than established ones, we use evidence from our survey to shed light on this point. In particular, in *Question Q2.1*, we ask developers the following question: “*How frequently do you use Generative AI tools (e.g., ChatGPT, Bard) in your software development workflow?*”. We use this question to study potential differences in the use of GAI between two groups of developers (which we created based on their self-reported experience): those with 1–3 years of experience (i.e., relatively new developers) and those with more than 3 years of experience (i.e., experienced developers). It can be seen from Figure D9 that we find no evidence for any differences in the frequency of GAI use between new and experienced developers. We also replicate this analysis using answers to *Question Q2.2* in our survey, which asks about their time of adoption: “*When have you started using Generative AI tools (e.g., ChatGPT, Bard) in your software development workflow?*”. Again, we find no evidence for a statistical difference between new and experienced developers (see Figure D10).