AI Agents and Higher-Order Work

Suproteem K. Sarkar
University of Chicago

March 22, 2026

Please click here for the latest version

How do AI agents influence knowledge work? This paper finds that agents shift worker effort from implementation to supervision, which especially benefits verifiable work and expert workers. I use data from the coding platform Cursor to study agents in software production. First, I find that workers have become less likely to produce output manually after AI agents were introduced, and are more likely to delegate work to agents. Second, workers use agents for abstract, higher-order tasks like delegation, context gathering, and planning. Third, agents are used more frequently in settings with easier-to-verify work outputs. Fourth, experienced workers appear more skilled at delegating work to agents—they ask fewer questions, seek more alignment through planning, and accept agent outputs at higher rates. Fifth, using variation from the platform's feature release timeline, I find that agents increase software output, especially for firms with more verifiable work and more experienced workers. The complementarity between AI agents and expertise contrasts with evidence on non-agentic AI tools, which workers are more likely to use for support than for delegation. Taken together, the results suggest that the returns to expertise may rise as knowledge work becomes more abstract.

First version: November 6, 2025. This paper was previously circulated with the title "AI Agents, Productivity, and Higher-Order Thinking: Early Evidence From Software Development." I thank Marianne Bertrand, Michael Blank, Jiafeng Chen, Jacob Conway, Joshua T. Dean, Joshua D. Gottlieb, David Holtz, Anders Humlum, Emir Kamenica, Lawrence F. Katz, Dennis Lee, Jacob D. Leshno, Yueran Ma, Luke Melas-Kyriazi, Amil Merchant, Max Miller, Virginia Minni, Sendhil Mullainathan, Dominic Russel, Karl Oskar Schulz, Joshua Schwartzstein, Rohan Shah, Andrei Shleifer, Adi Sunderam, Mirac Suzgun, and seminar participants at the University of Chicago and Yale. Part of this research was conducted while I was a visiting researcher at Anysphere in the summer of 2025. Some companies that shared data for this study had a limited right to review the manuscript prior to its release, in order to identify specific instances of confidential or personally identifiable information. I had editorial discretion over the paper and the right to publish.
Sarkar: University of Chicago, Booth School of Business (suproteem@uchicago.edu)

New technologies reshape how firms do work (David, 1990). These effects are uneven. Technologies impact some production processes more than others, which influences adoption and output (Atkinson and Stiglitz, 1969; Bartel et al., 2007). Technologies also reallocate the tasks workers perform, which influences the returns to worker skills (Katz and Murphy, 1992; Autor et al., 2003; Acemoglu and Restrepo, 2019).

AI agents integrate the general technology of a foundation model into the specific context of a firm. Early applications of AI provided workers with support for predetermined tasks or work in progress (Friedman, 2021; Brynjolfsson et al., 2025b; Angelova et al., 2025). Recent advances in agentic AI allow workers to delegate entire tasks to machines (Yao et al., 2023; Schick et al., 2023; Ide and Talamàs, 2025).

Agents change AI from a support tool to a delegation tool. The resulting shift in worker effort from implementation to supervision increases potential output. However, it also creates an alignment problem—agent actions may not follow worker intent (Aghion and Tirole, 1997). The alignment problem may be less severe when agents are used for easier-to-verify work (Holmström, 1979), and when workers can more clearly specify how to produce the output (Simon, 1973). Agents may therefore be more effective technologies for verifiable work and expert workers.

This paper analyzes data from the coding platform Cursor to study the impacts of AI agents on software production. Agents have replaced AI autocompletion as the primary AI feature used on the platform. The majority of messages from workers to agents delegate task implementation, though workers also use agents to gather context and develop plans. Agent usage is frequent in non-engineering functions, which are likely to have easier-to-verify software outputs. In addition, experienced workers appear to be more skilled at using agents. Based on variation from the agent feature's release timeline, I estimate that agents increased firm-level output, especially for firms with more non-engineers and firms with more experienced workers.

Background The current generation of AI agents combines foundation models with scaffolds—integrations that call external tools, and use feedback from tool outputs to guide model generations (Yao et al., 2023; Schick et al., 2023). In software development, these integrations allow agents to read code, browse company communications, search the internet, execute software, and edit code. In law, agents can search case databases, read internal documents, and edit drafts; in accounting, agents can read transaction data, flag anomalies, and generate financial statements. While workers can use standalone foundation models to ask general knowledge questions, agents introduce context from a

specific production environment, and actuators that allow workers to generate output within the production environment.

Agents let workers shift effort from implementation to supervision. Much of production is bound by implementation-heavy, *syntactic* tasks—work that translates specified intent into output. These tasks include writing code from a product roadmap, drafting slides from an outline, and matching expenses to accounting categories. As agents lower the cost of implementation, production shifts to higher-order, *semantic* tasks—work that specifies what to produce and evaluates generated output. These tasks include writing product roadmaps, producing outlines, and defining accounting categories—and evaluating the resulting output.

Software has been one of the earliest domains to adopt agents. In software, code is available in open-source codebases as pre-training data and is cheap to produce and reward as post-training data. Software also has a production feedback loop—developers of coding agents consume the tools they produce, which increases the pace of agent development. Moreover, software is a technical field whose workers may be more willing to use technologies at early stages.

The datasets used in this paper cover professional users of the software platform Cursor. The main analysis sample studies usage patterns using internal data covering 119,960 users across 1,000 randomly sampled firms. This sample is combined with data on usage frequencies, AI accept rates, and message contents. An additional sample of firm-level metadata covering 323,589 code merges from a subset of 32 firms that used a specific platform feature is used in the final section to estimate effects on output.

Agents shift work effort from implementation to supervision Agents have replaced AI autocompletion as the most common AI instrument used for software production. In the early 2020s, commercial software platforms like Copilot and Cursor offered AI autocompletion products, which suggest new code as users type into code editors. In late 2024, commercial software platforms began to integrate agents. While autocompletion offers suggestions as workers implement tasks, agents allow workers to delegate self-contained tasks. Cursor's agent was initially released in beta on November 24, 2024, with a full release on February 19, 2025. In April 2025, agents passed autocompletion as the most used AI instrument among active users. By the start of 2026, 92% of active users used agents.

In their messages to agents, workers specify tasks to implement, ask questions, and develop plans. 59% of agent messages are instructions to write code—messages with

implementation intents require workers to specify production guidelines and evaluate generated output. 33% of messages are questions to the agent, which include requests to explain code behavior and errors—workers delegate some information-gathering tasks to agents. 4% of first messages are instructions to develop plans—for complex tasks, generating and reviewing an agent-written plan may improve the alignment between user and agent (Mozannar et al., 2025).

Meanwhile, fewer workers appear to be executing tasks manually. By the end of the sample period, 57% of active users used AI autocompletion. Since manual code keystrokes in the code editor automatically trigger the AI autocompletion feature, as many as 43% of users may no longer be manually typing code. These results are consistent with a shift away from manual implementation and toward worker-supervised implementation with agents in software development.

Agents appear more effective for verifiable work and expert workers When agents reduce the cost of implementation, production becomes more bound by verification. Since agents may not perfectly adhere to worker intent, agent use becomes more feasible when outputs are easier to verify. Across roles, design mockups and product demos may be easier to verify than backend software and model training pipelines. Across sectors, consulting outputs like dashboards may be easier to verify than dependency-heavy outputs like developer tools and trading algorithms.

On the extensive margin, workers in roles and sectors with verifiable work outputs are more likely to use only agents, without any AI autocompletion usage. In an average week, 56% of designers and 61% of product managers use only agents, compared to 36% of software engineers and 35% of data/machine learning workers. In an average week, 40% of workers at consulting firms use only agents, compared to 38% of workers at financial firms and 37% of workers at software firms. Similar patterns hold on the intensive margin. Designers send 37% more agent messages than software engineers. Workers at consulting firms send 12% more messages than workers at financial firms, and 4% more messages than workers at software firms.

Agents may produce more aligned output for workers who can clearly specify intent and effectively coordinate with agents. These skills may be more developed among expert workers. To evaluate these predictions, I use work experience as a proxy for expertise. Experienced workers have spent more years gathering production context and delegating tasks to others, which may increase their proficiency using agents.

Experienced workers ask fewer questions with agents, and are more likely to develop

plans in their first messages. One standard deviation increase in work experience is associated with 4% lower ask intent rates across all agent messages, and 11% higher plan intent rates in conversation-starting messages. These relationships are quantitatively similar after adjusting for role and sector fixed effects. Experienced workers seem to have enough production context that they rely less on agents to answer questions. Experienced workers also seem to seek greater alignment with agents by developing and reviewing plans in conversation-starting messages.

Instructions from experienced workers appear to produce more aligned agent output. One standard deviation increase in work experience is associated with a 2% higher rate of accepting agent-generated output. By contrast, one standard deviation increase in work experience is associated with a 4% lower rate of accepting autocompletion-generated output. Autocompletion provides generic output suggestions—accepting these suggestions can break flow, which may be more costly for experienced workers. Meanwhile, agents allow workers to specify custom outputs, which may reward experienced workers who have more specification skill. This reversal of the experience gradient between AI autocompletion and agents mirrors the predictions from Ide and Talamàs (2025) on the differing effects of co-pilot and co-worker AI tools.

Agents increase output, especially for firms with verifiable work and expert workers
Agent use is associated with higher output. I use a difference-in-differences design that compares outcomes in the weeks surrounding the agent's release. When the agent was fully released on February 19, 2025, it became the default code generation tool on the platform. I analyze the output metrics sample, where the comparison is between 24 organizations that were already using the platform when the analysis period began (the "eligible group") and 8 organizations that were not using the platform during the analysis period (the "baseline group"). The data for both groups is backfilled and comprehensive—it does not restrict only to merges made through Cursor. In the 15-week period after the agent's full release, weekly code merges were 39% higher in the eligible group relative to time trends in the baseline group. There were no significant changes in revert rates as well as decreases in bugfix rates after the feature release—these results are consistent with no short-run codebase deterioration, but do not measure structural, longer-horizon changes in quality from agent use. In addition, there were no significant changes in lines edited per merge or files touched per merge, which is consistent with similar scope of additional output after the agent's release.

Output increases were greater in firms with smaller engineer shares and with more

experienced workers. I extend the difference-in-differences specification by splitting the eligible firms across medians of software engineer share and average work experience. For firms with lower software engineer shares, the increase in weekly merges was 50%; for firms with higher software engineer shares, the increase was 30%. For firms with lower average work experience, the increase in weekly merges was 23%; for firms with higher average work experience, the increase was 52%. In firms with many non-engineering workers like product managers and designers, work output may be easier to verify; in firms with many experienced workers, agent users may have more production context that guides more effective use.

Related literature This paper studies how AI agents influence knowledge work. Agents differ from previous applications of AI to production in two key ways. First, rather than providing workers with scoped recommendations and task guidance (e.g. Angelova et al., 2025; Friedman, 2021; Brynjolfsson et al., 2025b), agents allow workers to specify and delegate custom tasks. The shift from support to delegation may reward expert workers. Second, rather than operating as standalone foundation models (e.g. Noy and Zhang, 2023; Humlum and Vestergaard, 2025; Chatterji et al., 2025), agents use scaffolds that integrate models with specific production context. Agent scaffolds act as complements to the upstream technology of foundation models, which allow agents to be used both for context gathering and implementation.

This paper builds on the literature studying technology and production. The impacts of information technologies vary across task types, management practices, and organization characteristics (Mansfield, 1961; Bresnahan and Trajtenberg, 1995; Brynjolfsson and Hitt, 2000; Athey and Stern, 2002; Autor et al., 2003; Acemoglu and Autor, 2011; Syverson, 2011; Bloom et al., 2014; Acemoglu et al., 2024). Computation can scaffold memory and reasoning, and can offload mental processes to external environments (Hutchins, 2006; Kirsh and Maglio, 1994; Zhang and Norman, 1994; Clark and Chalmers, 1998; Risko and Gilbert, 2016). This paper studies the integration of coding agents into software production. Agents help to offload the syntactic activity of implementation, and shift work patterns toward communicating logic in natural language and evaluating generated outputs.

This paper contributes to the literature on AI, work experience, and expertise. Prior research that analyzes ChatGPT usage has found a junior skew—Humlum and Vestergaard (2025) find that the platform was more likely to be used by younger workers; Chatterji et al. (2025) find that younger people are a higher share of users. Brynjolfsson

et al. (2025b) find that a conversational assistant disproportionately improved output for lower-skill workers. Similar to this paper's findings on experience, Daniotti et al. (2026) find that experienced developers in open source software had higher AI-assisted commit rates than less experienced developers. In addition, Weidmann et al. (2025) experimentally find that skill leading human teams is related to skill leading AI agents. Labor market effects of AI tools depend on how applications relate to worker expertise (Garicano and Rossi-Hansberg, 2006; Autor and Thompson, 2025), changes in demand for worker skills (Katz and Murphy, 1992), and reallocation of tasks within roles (Hampole et al., 2025).[1] This paper finds that while less experienced workers are more likely to accept AI autocompletion suggestions, experienced workers are more likely to accept agent suggestions. In addition, agents increase output more in firms with more experienced workers. This evidence is consistent with the predictions from Ide and Talamàs (2025) and Autor and Thompson (2025), who find that the complementarity between automation and expertise depends on the distribution of automated tasks.

This paper also contributes to the literature on AI's effects on software development. One segment of this literature studies the effects of AI autocompletion usage, or treatments that mix autocompletion, non-agentic chat, and agent usage (Cui et al., 2024; Peng et al., 2023; Hoffmann et al., 2024; He et al., 2025; Chen and Stratton, 2026).[2] Recent empirical studies focusing on coding agents have analyzed the distribution of user messages (Anthropic, 2025) and the overall adoption of agents (Albarran, 2025). Chen et al. (2025) conduct a user study of agents with 20 student programmers and find increases in speed and correctness, as well as decreases in perceived understanding. Becker et al. (2025) randomize AI availability across tasks among 16 open-source developers and find longer completion times for AI-use tasks, and Shen and Tamkin (2026) randomize AI availability across 52 developers who apply a new software library,

[1] Recent hiring for junior workers has differed across occupations and firms—some researchers argue these differences may stem from AI exposure (Brynjolfsson et al., 2025a; Lichtinger and Maasoum, 2025), and other researchers find smaller aggregate reallocation (Gimbel et al., 2025). The effectiveness of AI applications has varied across tasks (Dell'Acqua et al., 2023; Dillon et al., 2025), and people do not always effectively infer how model performance generalizes across domains (Vafa et al., 2024).

[2] Field experiments (Cui et al., 2024) and task-specific experiments (Peng et al., 2023; Paradis et al., 2024) find shorter task completion times for people who use AI-assisted programming, particularly among less experienced developers. Across open source repositories, Hoffmann et al. (2024) find that developers eligible for AI-assisted coding tools shifted their work toward coding outputs and away from project management outputs, He et al. (2025) find transient increases in output and long-run increases in code complexity, and Daniotti et al. (2026) find increases in output, especially for experienced workers. Across professional developers, Chen and Stratton (2026) study productivity effects of AI-assisted programming tools among categories of output tasks, and Arcolano (2025) finds positive correlations between AI use, software production speed, and bugfix rates.

and find lower conceptual understanding among AI users. In interviews with more than 70 software developers, Thompson (2026) finds that programmers have been moving toward more abstract, supervisory work. This paper studies the diffusion and usage of agents across professional developers in 1,000 firms, and analyzes how agents influence production. By measuring message intent and output accept rates, and linking these to user characteristics, the paper provides micro evidence on how different types of workers use agents. It also distinguishes AI for support and delegation—and studies mechanisms like verifiability and expertise—which may help to interpret AI's heterogeneous effects on software development.

Finally, this paper contributes to the broader literature on AI's economic effects, which includes work studying AI and productivity (e.g. Noy and Zhang, 2023; Brynjolfsson et al., 2025b), the growing footprint of AI use (e.g. McElheran et al., 2024; Bick et al., 2024; Humlum and Vestergaard, 2025; Chatterji et al., 2025), and interaction patterns with language models (e.g. Zhao et al., 2024; Handa et al., 2025; Tomlinson et al., 2025; Chatterji et al., 2025).[3] This paper studies how AI agents influence the production of knowledge work.

1. Organizing Framework

I present a simple framework to organize the paper's empirical results. The motivating idea is that agentic AI differs from prior AI applications because it shifts AI's role from a support tool to delegation tool. When a worker delegates implementation, their role shifts to specifying the problem and verifying the result. This creates scope for higher output, but also introduces an alignment problem where output does not match worker intent. This alignment problem is reduced when output is easier to verify and workers are more skilled at specifying what they want. The framework predicts that workers use agents more for verifiable outputs, that expert workers develop more plans with agents, and that expert workers can use agents to produce more aligned output.

1.1. Setting

Each worker i has expertise $e_i \geq 0$: Higher expertise corresponds to higher domain knowledge and evaluation skill. Each work output j has verifiability $v_j \in [0,1]$: Higher

[3]Additional work in this area studies AI as cognitive automation for research (Korinek, 2023), market effects of agents (Rothschild et al., 2025; Shahidi et al., 2025; Hadfield and Koh, 2025), worker preferences for agents (Shao et al., 2025), and agents as representations of people (Liang, 2025; Imas et al., 2025).

verifiability means it is easier to evaluate whether the output satisfies the worker's goals. The value of a successfully completed output is normalized to one. The worker can complete the output using one of three modes

Manual implementation The worker implements the output directly. The worker's cost is

$$C^M(e_i) = \frac{1}{1+\phi e_i}$$

Copilot assisted implementation AI assists with local implementation but does not complete the full output. The cost is

$$C^C(e_i) = \frac{1-\alpha}{1+\phi e_i} + \kappa(e_i)$$

where $\alpha \in (0,1)$ is the direct implementation assistance by the AI copilot, and $\kappa(e_i)$ is an interruption cost. If experienced workers are already proficient at implementation, or if generic suggestions are less useful for them, the gain from copilot use does not necessarily increase in expertise.

Agent delegation The worker does not implement the output manually. Instead, the worker chooses specification effort $s \geq 0$, delegates the work to the agent, and verifies the output. The expected cost is

$$C^A(v_j, e_i) = \min_{s \geq 0} \psi(s, e_i) + \tau(v_j, e_i) + p(s, v_j)R$$

$R > 0$ is the cost of rework to fix a misaligned output. $\psi(s, e_i)$ reflects the cost of writing a good plan or specification. Assume $\psi(s, e_i) = \frac{s^2}{2(1+\lambda e_i)}$ with $\lambda > 0$, so specification is less costly for expert workers. $\tau(v_j, e_i)$ reflects the cost of verification. Assume $\tau(v_j, e_i) = \frac{\bar{\tau}(1-v_j)}{1+\mu e_i}$ where $\bar{\tau}, \mu > 0$, so verification is easier for verifiable outputs and expert workers. $p(s, v_j)$ reflects the probability of misalignment between worker intent and agent output. Assume that $p(s, v_j) = \exp\{-(\beta_s s + \beta_v v_j)\}$ with $\beta_s, \beta_v > 0$.

This setup makes a distinction between support tools and delegation tools. Copilots reduce the cost of implementation while the worker remains the main producer. Agents reduce the need for implementation, while increasing the importance of specification and verification. The comparative advantage of an expert worker depends on which

human tasks remain after an AI tool is introduced.

1.2. Predictions

The worker chooses the production mode with the lowest expected cost. Agent delegation is optimal when $C^A(v_j, e_i) \leq \min\{C^M(e_i), C^C(e_i)\}$. Section A.1 presents additional derivations supporting the predictions that follow from this setup.

Prediction 1 (Agent delegation is more attractive for outputs with higher verifiability)**.** *For fixed e_i, $C^A(v_j, e_i)$ is strictly decreasing in v_j, while C^M and C^C do not depend on v_j.*

Prediction 2 (Optimal specification effort rises with worker expertise)**.** *The first-order condition is $\frac{s^*}{1+\lambda e_i} = \beta_s R \exp\{-(\beta_s s^* + \beta_v v_j)\}$. Implicit differentiation gives $\frac{\partial s^*}{\partial e_i} > 0$.*

Prediction 3 (Conditional on delegation, expertise raises alignment)**.** *Under the baseline assumptions and the interiority result, alignment rises with expertise: $\frac{d}{de_i} \log p\big(s^*(v_j, e_i), v_j\big) < 0$. This follows from $\frac{d}{de_i} \log p\big(s^*, v_j\big) = -\beta_s \frac{\partial s^*}{\partial e_i}$ and the previous prediction.*

Prediction 4 (A delegation tool can increase the return to expertise even if a support tool does not)**.** *Define the return to expertise in mode $m \in \{C, A\}$ as $-\frac{\partial C^m}{\partial e_i}$. Delegation has a strictly larger return than copilot use if and only if expertise matters more for the supervisory parts of production versus the implementation parts (full condition in Section A.1). If expert workers are already strong at implementation, or if generic suggestions are less useful for them, the gains from copilot use do not necessarily increase in expertise. Meanwhile, delegation can raise returns to expertise through the specification and verification channels.*

Similar logic applies to firm-level outcomes. If output decreases in expected completion cost, then the firm-level output gain from agents can be larger in firms with more high-v work outputs and more high-e workers.

2. Context and Data

I describe software development and AI-assisted programming, and document the data sources used in this paper.

2.1. Software Development and AI-Assisted Programming

Software development process Software development follows a lifecycle similar to product development in other contexts. An organization identifies problems, proposes

solutions, and designs specifications for new software. Software developers write code to implement these designs, and test and verify the output of these code changes. Once software is finalized, it is deployed and maintained by each organization. Modern development proceeds iteratively, with teams revisiting earlier stages of the lifecycle throughout.

To implement software, developers often use code editors. Code editors are programs that integrate text editing with other features. Feature-rich code editors can support debugging, error detection, and version control integration. In addition to code editors, developers also produce software using command-line tools, visual programming environments, and other code generators.

Organizations track code changes using version control, and apply both manual and programmatic review before deploying proposed changes. Version control systems record modifications to a codebase over time. Once a developer has finished a work item, they typically submit their code for review by other developers and automated verifiers. Reviewed code changes are then merged into the main codebase.

Once code changes are reviewed and merged, organizations deploy the overall codebase to production. With the growth of continuous integration and continuous deployment pipelines, code changes are being deployed to production at increasing frequency. If a deployment results in a failure—for example, through degraded service or an outage, it is typically rolled back or hotfixed by the organization.

Software development productivity is typically measured using metrics related to speed, quality, system health, collaboration, and satisfaction. Forsgren et al. (2019) discuss metrics related to development speed (deployment frequency, lead time for changes), and development health (change failure rate, time to restore services). Forsgren et al. (2021) discuss holistic benchmarks that measure satisfaction, performance, activity, communication, and efficiency.

AI and software development Software development has been a frequent domain for AI applications. Code includes repetitive patterns (Hindle et al., 2016) that language models can recognize and apply. Many software outputs can be verified with tests, and such verification is useful for training language models for software development (Chen et al., 2021; Le et al., 2022; Chen et al., 2022). Modern software development also makes extensive use of code review and continuous integration (e.g. Martin, 2012; Sadowski et al., 2018; Fowler, 2024)—these validation practices may control the quality of AI-assisted code that ultimately gets deployed. In addition, people who produce AI-

assisted programming platforms may regularly use such platforms as they work, which can speed up the production feedback loop.

Autocompletion and generation Two common modalities of AI-assisted programming are autocompletion and generation. Autocompletion is triggered by code editing keystrokes, while generation is triggered by natural-language messages.

Autocompletion suggests new code as users type into a code editor. Early autocompletion tools performed completion of matching code phrases—for example, a user could press tab to finish typing a variable name previously defined in the codebase. Subsequent tools performed language model-based autocompletion, which can generate code sequences that may differ from existing patterns in the codebase.

Generation produces code from natural-language conversations with users. Early generation tools, like program synthesizers, transformed high-level specifications into simple programs. Subsequent tools performed richer generations that built on language model outputs. With a natural language interface, a user does not need to type in specific initial code sequences for a generation tool to produce code.

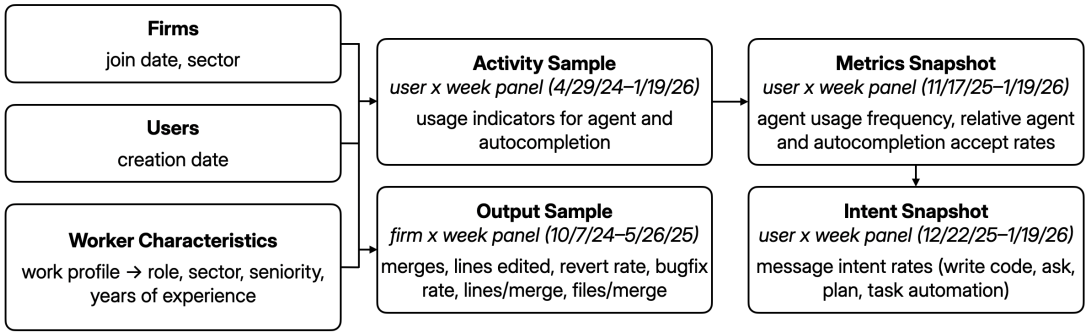
Agentic generation Recently, generation tools have implemented agentic capabilities. An agent takes a natural language instruction from a user and generates code by combining a language model with tools that can read files, run tests, access APIs, and conduct other tasks. Agents can break up user instructions into subtasks, and interact with external tools to generate code changes.

In addition to producing code edits, agents produce chat that is tied to a specific codebase. While standalone language models can produce chat using general knowledge, agents integrate language models with additional context. Workers can use agents' codebase integrations to ask targeted questions. The back-and-forth interaction pattern with agents also gives workers the opportunity to generate and review agent-written plans.

2.2. Data

This paper uses data on user characteristics, usage patterns, and software output across professional users of the coding platform Cursor. Figure 1 describes the major data sources.

Figure 1: Summary of data sources and sample coverage.



Notes – This figure describes the main data sources used in the paper. The activity sample consists of activity indicators across 1,000 firms. The metrics snapshot filters to weeks with available data on AI usage, and the intent snapshot further filters to user-weeks which used a feature that generated labels of message intent. The output sample consists of output metrics across 32 firms which joined in the weeks surrounding the sample and used a feature that generated labels of firm-level output.

User characteristics and agent adoption I analyze aggregated usage metrics and professional characteristics from a random sample of 1,000 firms. The sample is uniformly drawn from a subset of firms with 50 or more users, with a recorded sector, and which had started using Cursor before July 15, 2025. The sample consists of 119,960 users affiliated with these firms who joined the platform before September 1, 2025 and were matched to role, seniority, and years of work experience. Sector is an internal Cursor designation at the firm level. Professional characteristics were obtained by using a language model classifier to label de-identified work history summaries. Section A.2 contains details on the labeling procedure. I measure adoption using the share of weekly users who used agents, autocompletion, or both tools ranging from the week of April 29, 2024 to the week of January 19, 2026.

Usage frequency, accept rates, and conversation intent I analyze data on weekly total agent requests, agent accept rates, autocomplete accept rates, and agent message intent using a snapshot of telemetry metrics collected from the week of November 17, 2025 through the week of January 19, 2026. The agent accept rate is the share of messages with recorded edit actions for which a user accepted a code edit. The autocomplete accept rate is the share of suggested tab autocompletions that a user accepted. Accept rates are collected in relative terms, scaled by the sample mean. Agent message intent is collected from a Cursor service used by a subset of organizations. The

service classifies the intent of a user message into "Write Code," "Ask," "Plan," "Task Automation," and a small share of other categories. Message intent data is available for 399 organizations between the weeks of December 22, 2025 and January 19, 2026. Table A3 shows that usage frequency is 14% higher in the subsample matched to intent classification, and the Appendix reports robustness exercises that filter the accept rate sample to the message intent subsample. For some analyses, I also use task category classifications generated by the same feature used to infer message intent.

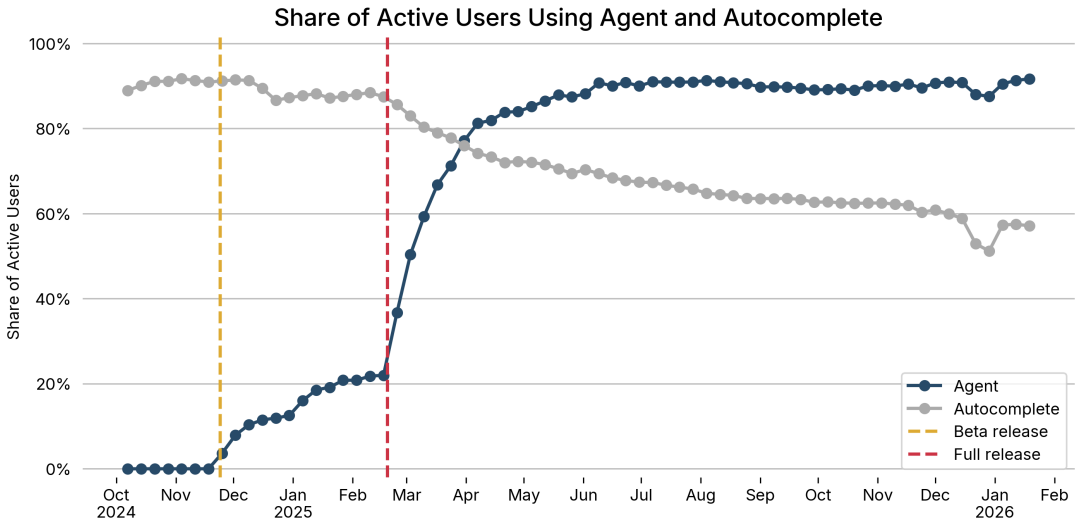
Output, quality, and scope proxies To measure aggregate effects of agents on output, I use a panel of organization-level merge metadata from October 2024 to May 2025. The metadata contains merge week, number of lines edited, number of files touched, and categorizations into reverts and bugfixes across 32 organizations with available aggregate merge metadata. I compare outcomes between an "eligible" group of organizations that started using Cursor before the analysis period began, and a "baseline" group of organizations that did not start using Cursor until after the analysis period ended. Since the eligible group was already using the platform when the analysis period began, its production technology would have been affected by the agent initial release and agent full release updates. Meanwhile, the baseline group's production technology would not have been affected by these updates during the analysis period. There are 24 eligible organizations and 8 baseline organizations, with 323,589 merges covered over the 34-week sample period. For both groups, this metadata is historical and comprehensive—it is not restricted to merges made using Cursor, and is backfilled across the sample period. To measure developer-level associations between agent usage and output, I analyze merge and deployment metadata from an organization referred to as Company 1. This metadata is used for cross-sectional analyses that relate agent usage styles to outcome metrics across developers.

3. The Shift to Agentic Production

Agents have become a central component of software production. The Cursor platform released the initial, beta version of its coding agent in November 2024, with a full release in February 2025. By April 2025, more active users used agents than used AI autocompletion. At the end of the sample period, 92% of active users used agents.

Agents change how workers produce output. Workers message agents using natural language, which involves a series of semantic tasks. One type of task is specification—in

Figure 2: Agents have become a widely-adopted AI code generation tool.



Notes – This figure plots the share of weekly active users who used the autocomplete feature or the agent feature. The agent feature was initially released in the November 24, 2024 update, and was made the default generation mode after its full release in the February 19, 2025 update. The agent would become the default setting when users updated their software clients.

an average week, 59% of worker messages are instructions to write code. A second type of task is evaluation—agentic code edits produce output for workers to verify. A third type of task is information gathering—33% of messages are questions, including questions about code and errors. A fourth type of task is alignment—since agents and workers may differ in their implementation strategies, workers may benefit from aligning the agent's output to their intent. 4% of initial messages to agents are instructions to develop plans, which may improve alignment when workers review agent trajectories before producing output (Mozannar et al., 2025).

3.1. Growth of Agent Usage

In the months following the agent feature's release, agents became the most common AI tool used on the platform. Figure 2 plots the weekly share of active users who used the agent and autocompletion features. The beta release of the agent was on November 24, 2024, and the full release was on February 19, 2025. After the full release, agents became the default code-generating tool for users who updated their software clients. At the end of the period between beta and full release, 21.8% of active users adopted the agent tool. In April 2025, agents surpassed autocompletion as the most used AI tool among active

users. At the end of the sample period during the week of January 19, 2026, 91.7% of active users used agents while 57.2% of active users used AI autocompletion.

What drove the shift to agentic production? Figure A4 plots the weekly adoption share across user creation cohorts. Endline agent usage was 92.4% for users who joined after the agent's full release, 90.5% for users who joined between beta and full release, and 86.1% for users who joined before the beta release. Endline AI autocompletion usage was 54.0% for users who joined after the agent's full release, 68.4% for users who joined between beta and full release, and 74.1% for users who joined before the beta release. Part of the decline in autocompletion usage reflects a composition effect—new users were less likely to adopt the older autocompletion technology. However, Figure A4 shows that part of the decline is also secular—early users who joined the platform before the agent feature's beta release also became less likely to use autocompletion.

Since autocompletion keystrokes are triggered by manual code edits, as many as 43% of users may not be coding manually in a typical work week. This figure represents an upper bound, as a fraction of users may disable the autocompletion feature (though the platform reports this is rare). In addition, some users may type code manually using alternative editors. As most major IDEs have agentic integrations in 2026, it would be unlikely for a large share of users to switch between IDEs for agent use and code editing, but it is feasible that some workers use a non-agentic editor like vim. Given these workers are likely to make up a small share of the platform's users, it is reasonable to assume that a meaningful share of users no longer types code manually. In any event, more than 90% of users now use agents in a typical week.

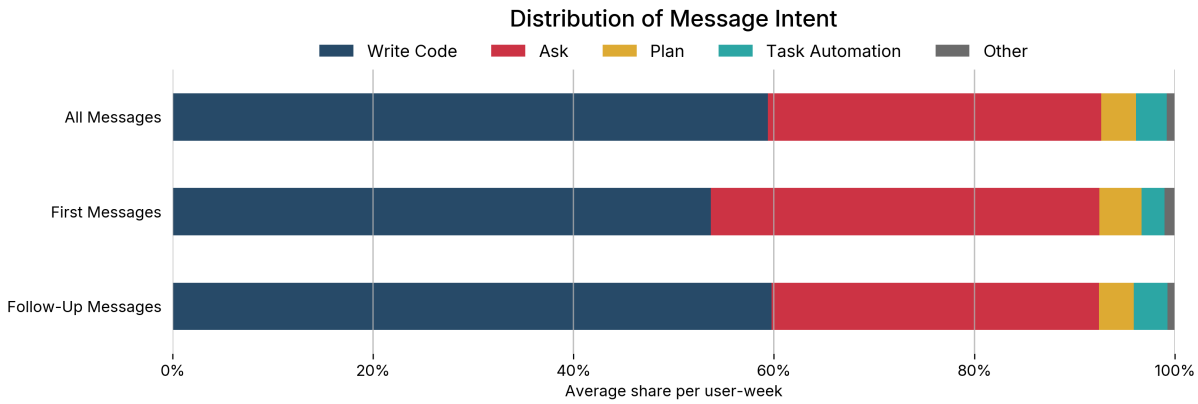
3.2. Higher-Order Tasks with Agents

Part of production involves lower-level, syntactic tasks—work that transforms specified intent into output. Part of production involves higher-order, semantic tasks—work that specifies intent and evaluates output. Agents may shift work effort from syntactic tasks to semantic tasks.[4]

Workers use agents both to implement and to gather information. After agents generate code, workers review the proposed changes and can choose to accept or reject. Some messages instruct agents to generate plans, which workers can review before

[4]The distinctions relate to *process tasks*, not *output tasks*. Output tasks are the artifacts that a worker produces. Hoffmann et al. (2024) and Chen and Stratton (2026) study how AI assistance may affect output tasks in software engineering. Process tasks are the procedures that workers use to produce output. Agents can change the distribution of a worker's process tasks even if the distribution of output tasks does not change.

Figure 3: Agent usage involves specification and evaluation.



Notes – This figure plots the average share of message intents across user-weeks. The first bar plots averages across all messages. The second and third bars plot averages for first messages and follow-up messages, respectively.

proceeding with implementation.

Figure 3 plots the average share of message intents across user-weeks in the agent intent sample. On average across user-weeks, 59.4% of messages have a "Write Code" intent, 33.2% have an "Ask" intent, 3.5% have a "Plan" intent, 3.1% have a "Task Automation" intent, and 0.8% have an "Other" intent.

Higher-order tasks associated with agent use include:

- *Specification.* Both "Write Code" and "Task Automation" messages require workers to specify their output goals to agents.
- *Evaluation.* After an agent generates code, users have the option to accept or reject the proposed changes. Work outputs like design mockups are more implementation-bound than verification-bound, and may benefit more from agents.
- *Information gathering.* Workers use agents to ask questions, including to explain code and to explain errors. While standalone language models can answer general knowledge questions, agents can answer questions tied to a specific codebase.
- *Alignment.* Complex outputs may require multiple implementation decisions. Given the same production idea, workers and agents may not make the same decisions. Planning messages let workers review agents' proposed changes before implementation, which may improve alignment between worker and agent.

First messages are less likely to have execution-oriented intents. For first messages across user-weeks, 53.7% have a "Write Code" intent, 38.8% have an "Ask" intent, 4.2% have a "Plan" intent, 2.3% have a "Task Automation" intent, and 1.0% have an "Other" intent. Workers are 17% more likely to have "Ask" intents and 20% more likely to have "Plan" intents in their first messages than in all messages.

4. Agents Benefit Verifiable Work and Expert Workers

Agent usage patterns across roles and work experiences suggest that agents benefit verifiable work and expert workers. The majority of agent users work in engineering roles. However, there is a mass of frequent usage in non-engineering roles—many of these workers only use agents, and do not use AI autocompletion. These worker may conduct easier-to-verify work that may be suited for agents. Experienced workers appear to be more skilled users of agents. They ask fewer questions, generate more plans, and accept agent outputs at higher rates. The positive experience gradient for agent accepts contrasts with a negative experience gradient for autocompletion accepts.

4.1. Worker Characteristics

I report summary statistics of worker characteristics. Figure 4 plots the distribution of users across the full sample. Table A2 reports additional summary statistics across subsamples studied in this paper.

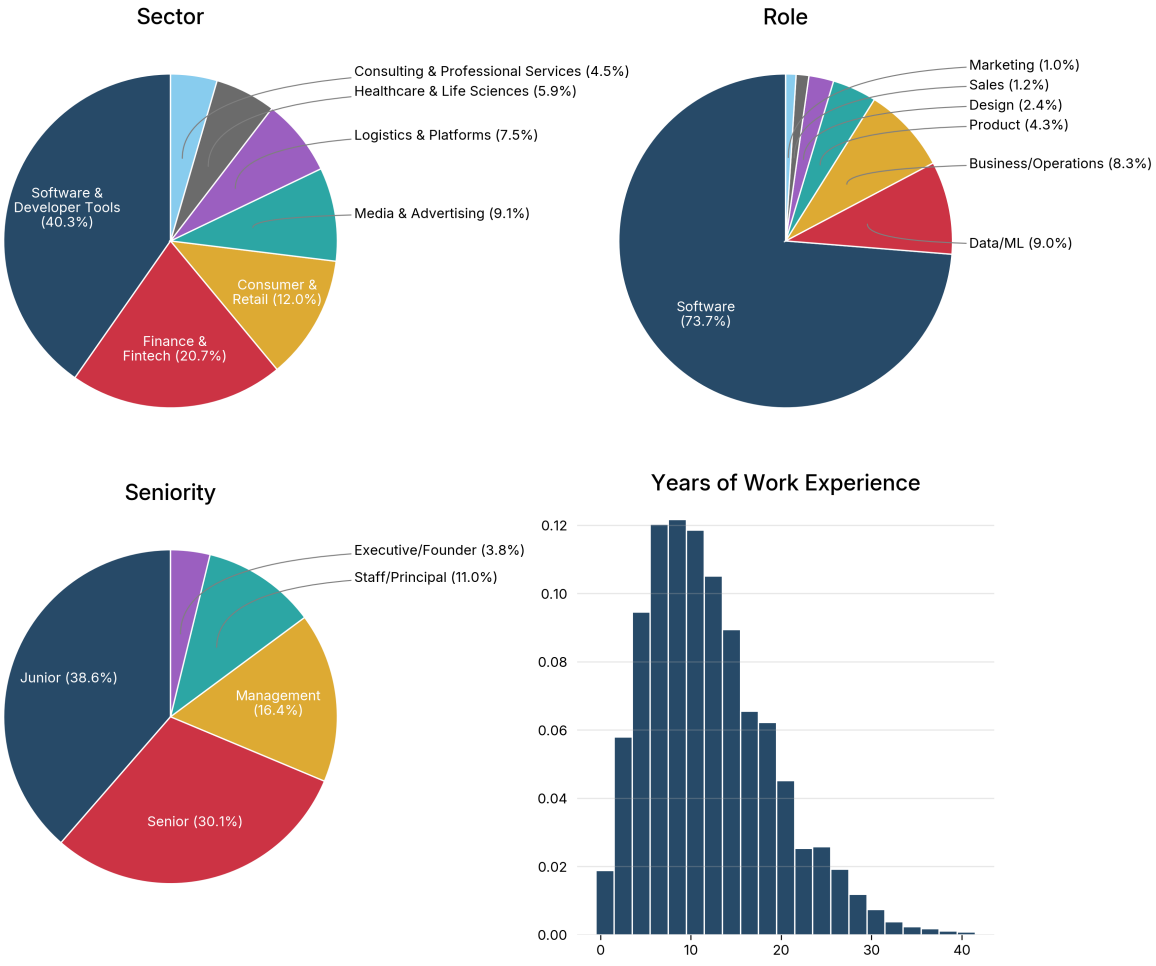
Workers across sectors use agents. 40% of workers in the sample work in organizations that produce software and developer tools. A further 42% work in finance/fintech, consumer/retail, and media/advertising (which includes gaming).

While the majority of workers hold software roles, the platform is also used in roles with traditionally lower programming intensity. Software roles account for 74% of workers in the sample, with data and machine learning roles accounting for a further 9%. Additional roles include product management, design, and sales.

39% of workers hold Junior roles, 30% hold Senior roles, and 11% hold Staff/Principal roles ("Staff" and "Principal" are more senior roles in software development). A further 16% hold management roles, while 4% hold founder and executive roles. Most users have 5–15 years of work experience.

For junior workers, the mean years of work experience is 8.5 and the standard deviation is 5.7. The corresponding statistics are 12.1 (5.9) for senior workers, 16.1

Figure 4: Worker characteristics.



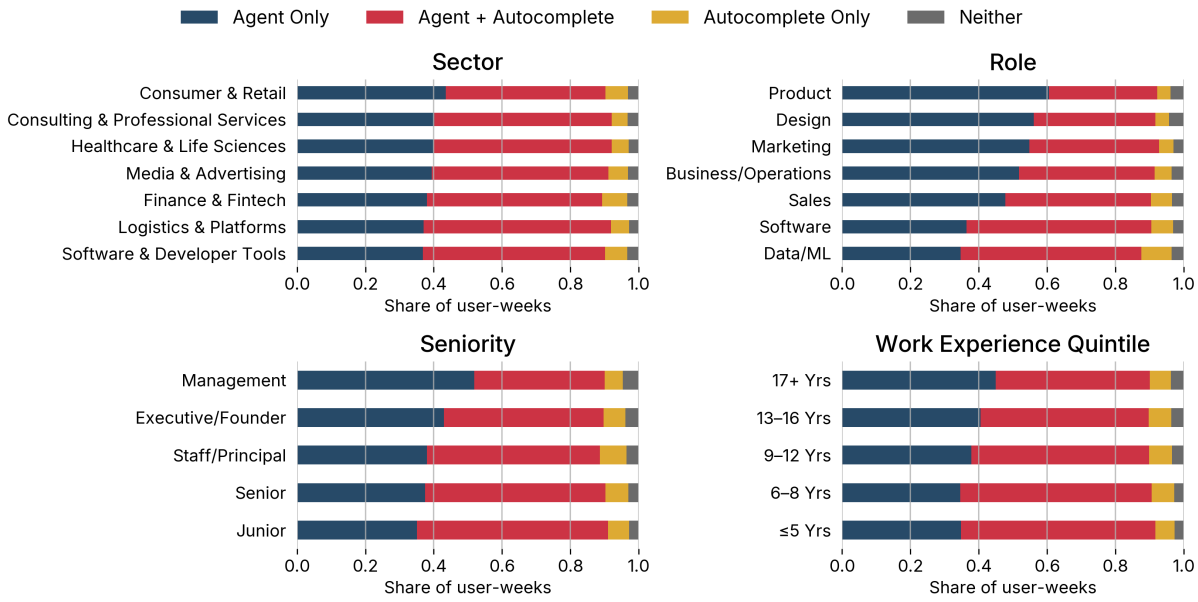
Notes – This figure plots the distribution of workers in the sample across sectors, roles, seniorities, and years of work experience.

(6.7) for staff/principal workers, 16.2 (6.5) for management workers, and 18.5 (7.9) for executive/founder workers.

4.2. Verifiable Work

Workers in non-engineering roles and sectors are more likely to use agents than they are to use AI autocompletion. They also use agents more often per week. Output tasks in these use cases include dashboards, design mockups, and one-off projects with few formal dependencies. These outputs may be easier to verify, which may increase the returns to agent use.

Figure 5: Extensive margin: Many workers only use agents.

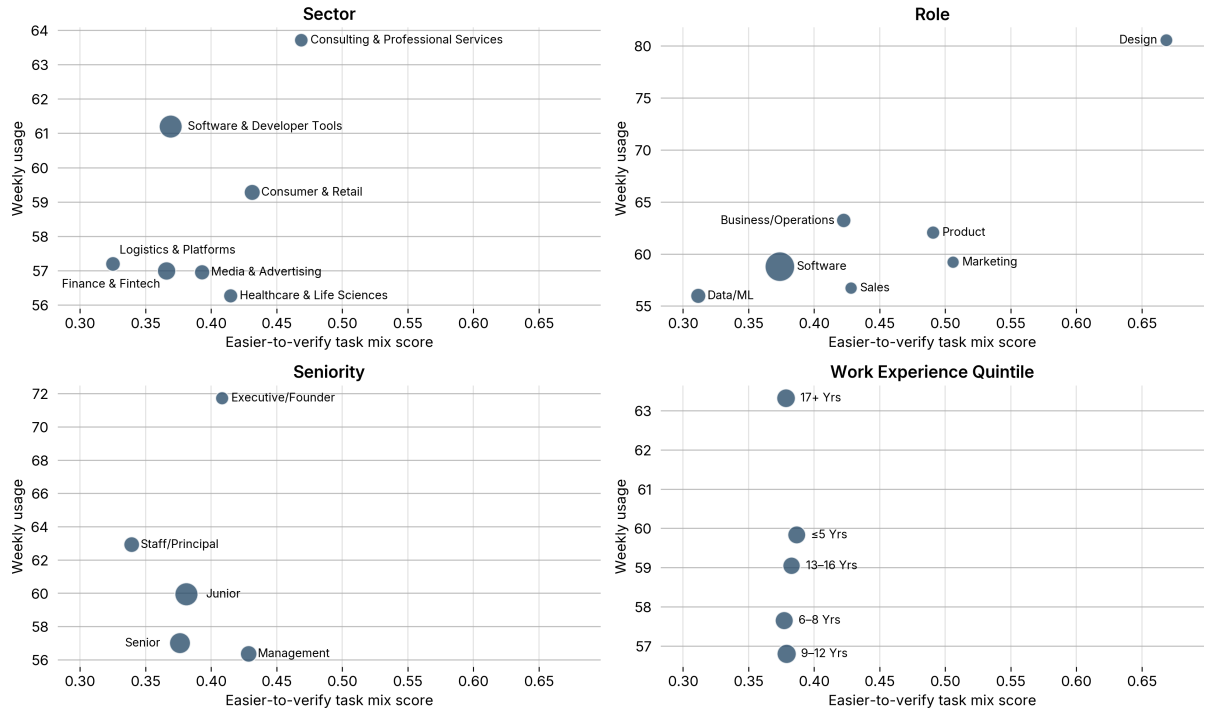


Notes – This figure plots the share of user-weeks with only agent usage, only autocompletion usage, or usage of both tools. Each subfigure splits the sample by sector, role, seniority, or quintile of work experience. The data derives from the agent metrics sample collection window from the week of November 17, 2025 through the week of January 19, 2026. The majority of user-weeks involve agent use, and many workers only use agentic tools.

Extensive margin: Many workers only use agents Figure 5 plots agent and autocompletion usage shares across worker characteristic groups. Agent-only usage is high in non-engineering sectors and roles. Workers in consumer, retail, and consulting firms are more likely to use only agents than workers in other sectors. The majority of product managers, designers, marketers, and operations specialists only use agents. AI autocompletion is more common across workers in financial firms, which may value precise implementation, as well as across data/ML workers.

Task distributions across roles Figure A5 plots the distribution of tasks across worker characteristics. These labels are obtained from the same Cursor feature used for message intent classification. Consistent with the view that non-engineering workers have easier-to-verify outputs, designers and product managers are more likely to use agents for UI or API integration tasks that have fewer formal dependencies across the codebase. Meanwhile, engineering workers are more likely to use agents for architectural and devops tasks, which suggest stronger dependencies across the codebase.

Figure 6: Intensive margin: Frequent agent usage in non-engineering roles.



Notes – This figure plots the average weekly agent requests across sectors, roles, seniorities, and quintiles of work experience. The horizontal axis reflects each worker characteristic's easier-to-verify task mix score. The score is the ratio between the average share of tasks that are related to UI/Styling or AI Integration as opposed to Architecture or DevOps/Deployment. Point sizes are proportional to the number of observations in each characteristic. The data derives from the agent metrics sample collection window from the week of November 17, 2025 through the week of January 19, 2026.

Intensive margin: Frequent usage by designers and other non-engineers Figure 6 plots weekly agent usage across worker characteristics. On the horizontal axis, each characteristic c is sorted by an "easier-to-verify task mix score" Ψ_c. This score is based on the characteristic-level task category taxonomy from Figure A5. It is defined as

$$\Psi_c = \frac{s_c^{\text{UI \& Styling}} + s_c^{\text{API Integration}}}{s_c^{\text{UI \& Styling}} + s_c^{\text{API Integration}} + s_c^{\text{Architecture}} + s_c^{\text{DevOps \& Deployment}}}$$

for each worker characteristic c. Here, s_c^T corresponds to the share of conversations with category T in characteristic c, averaged across user-weeks using task category data from the intent sample. This worker characteristic-level measure contrasts UI & Styling and API Integration tasks, which are consistent with easier-to-verify work, with Architecture

and DevOps & Deployment tasks, which are consistent with harder-to-verify work.

Designers are among the most frequent agent users. Design work involves tasks like creating mockups and live artifacts. Design work may be easier to verify through inspection than engineering work, which can involve more formal dependencies and heavier testing. Figure A6 finds similar results for agent-only usage. Consistent with the view that design work has fewer dependencies, Figure A7 shows designers also have the highest shares of "Write Code" and "Task Automation" messages, with fewer "Ask" and "Plan" messages.

Figure 6 also shows that workers at consulting firms are among the heaviest agent users. If engineers at consulting firms work on scoped projects for each client, there may be fewer dependencies required for each set of outputs. If this work involves dashboards or simple data analysis, outputs may also be easier to verify through inspection. Meanwhile, workers at finance firms are among the lightest agent users. If code outputs in this sector have more dependencies and require more effort to verify, workers may not be willing to delegate implementation of critical outputs to an agent.

4.3. The Experience Gradient

Experienced workers appear to be more skilled users of agents. Experienced workers ask fewer questions, which may indicate they already have more production context. They also plan more, which may improve alignment with agents. Experienced workers accept agent outputs at higher rates. This positive experience gradient for agent accepts contrasts with a negative experience gradient for autocompletion accepts.

Experienced workers plan more and ask less To evaluate the relationship between work experience and first message planning and asking intents, I estimate the specification

$$\text{Intent}_{i,t} = \alpha_t + \delta_{c(i)} + \beta \text{Experience}_i + \varepsilon_{i,t} \tag{1}$$

across workers i and weeks t. α_t is a week fixed effect and $\delta_{c(i)}$ encodes role and sector fixed effects. Years of experience are standardized to zero mean and unit variance.

Table 7 reports the results of these regressions on the share of "Ask" and "Plan" intents across first messages. Unconditionally, one standard deviation increase in work experience is associated with a 0.45 percentage point higher plan rate and a 0.88 percentage point lower ask rate. Compared to base rates, these reflect 11% higher plan

Table 7: Experienced workers are more likely to build plans, less likely to ask questions.

	First-Message Intent			
	Plan Rate [%]		Ask Rate [%]	
Work Experience (σ)	0.45***	0.42***	-0.88***	-0.85***
	(0.05)	(0.05)	(0.13)	(0.13)
Constant	4.21***		38.77***	
	(0.04)		(0.12)	
R^2	0.001	0.003	0.001	0.005
Within R^2	-	0.001	-	0.001
Observations	118,407	118,407	118,407	118,407
Week FE		✓		✓
Role FE		✓		✓
Sector FE		✓		✓

Notes – This table reports results of regressions of "Plan" and "Ask" message intent on work experience. The second column in each set adds week, role, and sector fixed effects. The data derives from the intent snapshot window from the week of December 22, 2025 through the week of January 19, 2026. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

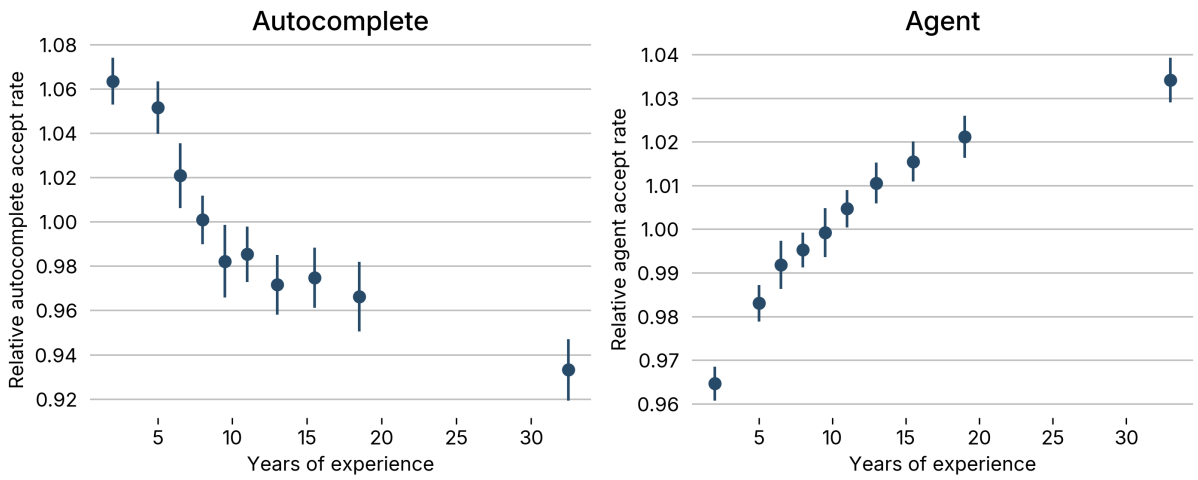
rates and 2% lower ask rates. Results are similar after adjusting for week, role, and sector fixed effects.

If experienced workers have more experience delegating tasks and more skill reviewing others' plans, they may be more likely to develop plans with agents before acting. If experienced workers have accumulated more production context, they may find it less useful to use agents to acquire information.

Table A9 and Table A10 report analogous results for first messages across all intent types, and Table A11 and Table A12 report analogous results for all messages across all intent types. Experienced workers are more likely to use agents for task automation, both across first messages and all messages. Without conditioning on first messages, experienced workers are even more likely to use agents to write code and even less likely to use agents to ask questions.

Experience gradient in accept rates Figure 8 plots the mean autocomplete and agent accept rates across deciles of work experience. Experienced workers are less likely to accept suggestions from AI autocompletion tools. This evidence is consistent with prior

Figure 8: Agents reverse the experience gradient in accept rates.



Notes – This figure plots autocomplete and agent accept rates across deciles of work experience. Both accept rates are scaled by their respective in-sample means. The data derives from the agent metrics sample collection window from the week of November 17, 2025 through the week of January 19, 2026.

work that finds a junior skew in AI tool usage.

What is different about agents? While traditional AI applications offer suggestions and guidance, agents let workers delegate self-contained tasks. Experienced workers may benefit less from suggestions than junior workers. Furthermore, if experienced workers have stronger preferences than junior workers, they may be less willing to accept generic autocompletion suggestions. However, agents allow experienced workers to specify specific tasks for implementation, which may increase experienced workers' willingness to use agentic tools. Higher skill specifying intent may then further increase the alignment between message and output, and increase accept rates for experienced workers.

The positive experience gradient in agent accept rates persists after adjusting for additional controls that may be related to worker task distributions. I estimate the specification

$$\text{Accept Rate}_{i,t} = \alpha_t + \delta_{c(i)} + \beta \text{Experience}_i + \varepsilon_{i,t} \tag{2}$$

across workers i and weeks t.

Table 9 reports the results of these regressions. After adjusting for role and sector, one standard deviation higher experience corresponds to 2.1% higher accept rates relative to

Table 9: Experienced workers have higher agent accept rates.

	Relative Accept Rate × 100				
	(1)	(2)	(3)	(4)	(5)
Work Experience (σ)	2.10***	2.10***	2.11***	2.13***	1.78***
	(0.05)	(0.05)	(0.05)	(0.05)	(0.06)
Constant	100.00***				
	(0.05)				
R^2	0.007	0.009	0.009	0.009	0.010
Within R^2	-	0.007	0.007	0.007	0.004
Observations	436,871	436,871	436,871	436,871	436,871
Week FE		✓	✓	✓	✓
Role FE			✓	✓	✓
Sector FE				✓	✓
Seniority FE					✓

Notes – This table reports results of regressions of relative agent accept rates on work experience. The columns progressively add week, role, sector, and seniority fixed effects. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

the mean. After further adjusting for job title seniority, which may over-control given its relationship with experience, one standard deviation higher experience corresponds to 1.8% higher accept rates relative to the mean. Table A15 reports the results of a regression restricting to the message intent sample, and finds similar results.

Interpreting accept rates and sample composition The decision to accept an agent generation depends on the alignment between agent output and user intent, the types of tasks a user conducts with agents, and a user's threshold for accepting generated code. Alignment may vary with a user's skill providing instructions to agents, including through clarity of description and effective mental models of agent capabilities. Task distributions may also vary—an alternative explanation for these results is that experienced workers perform tasks that are more suited for agents. Unconditional accept thresholds may vary with the value of aligned output to each individual. For example, less experienced workers may be newer team members who are more conservative acceptors of agent-generated output.

Further evidence is consistent with a skill-based explanation for the gap. A task

distribution-based explanation would imply that experienced workers perform tasks better suited for agents. However, Table 9 finds that the experience gap in accept rates persists after adjusting for role, sector, and seniority—the task distribution-based explanation would require selection on unobservables not explained by these characteristics. In addition, Figure 6 shows that the verifiable task mix score does not vary across years of work experience. A threshold-based explanation would imply that experienced workers have incentives or usage styles that drive up accept rates. If experienced workers conduct higher-value work with higher costs of failure, it would be unlikely for such a relationship to generate higher agent accept rates. If experienced workers were less vigilant agent users, they might be less likely to make plans in their initial messages, which would run counter to the results in Table 7. If experienced workers were less risk-averse, they might have higher agent accept rates than less experienced workers, but would be unlikely to have lower autocompletion accept rates as reported in Figure 8 unless risk perceptions differ between tools. It seems likely that experienced workers have more skill specifying delegated tasks and evaluating generated outputs.

These agent usage and accept statistics are conditional on a worker's choice to adopt the platform and submit messages to agents. Usage patterns may change as adoption patterns change. For example, while designers have the most frequent agent usage, designers who currently use AI-assisted programming tools may be more active in software development than most designers at software organizations. Nonetheless, these statistics correspond to usage patterns within the sample of adopting users. To measure firm-level effects, I turn to an analysis of firm-level outcomes.

5. Agents Increase Output

The release of the agent feature was associated with increases in software output. Section 5.1 reports firm-level difference-in-differences estimates across historical merge metadata from 32 firms. Using variation from a natural experiment induced by the agent feature's release timeline, I estimate effects of the agent release on merges, lines edited, revert rates, and bugfix rates.

Output increases were larger for firms with more non-engineers and firms with more experienced workers. Section 5.2 reports differences-in-differences estimates that split the eligible group by the pre-beta-release medians of software engineer share and average years of experience.

In addition, frequent agent users produce more output—they merge more code, edit

more lines, and edit more files per merge. Section 5.3 reports person-level correlational evidence on agent usage and software development outcomes from one organization.

5.1. Aggregate Effects on Output

I use variation from the agent feature's release timeline to estimate effects of agents on output. Figure 2 shows that the beta release and full release of agents both induced more workers to use agents. I use a difference-in-differences strategy that estimates effects relative to these releases.

Measurement Using the weekly panel of merge metadata described in Section 2.2, I analyze output and quality metrics in the weeks surrounding the release of the agent feature. The data corresponds to historical and comprehensive outcomes observed at the firm level.

As the main measure of output, I use the rate of weekly merges to the main codebase. Merges are units of work—a developer submits a merge request to address a specific goal. While merges can be interpreted as economic units of work, their impact is not uniform—some code merges may address more valuable business goals than others. However, since merges are units of code output used by all organizations, they allow for aggregate analysis of code output.

As an additional measure of output, I include weekly lines edited. Lines edited per merge are winsorized at the 1% level at the right tail, and then summed across merges each week. Edited line count is a syntactic metric, and varies with programming language verbosity. It may also increase if agents produce more code comments or more verbose code for similar logic. Like merges, lines edited are units common to all organizations.

As the main proxy for quality, I use the weekly share of merges that are labeled as reverts. The revert rate is a proxy for the frequency of breaking code changes across a codebase. An advantage of this metric is that it indicates a likely severe issue—for example, an organization may revert a merge if it creates degraded service in deployment. However, the revert rate may oversample high-visibility, short-horizon issues.

As an additional proxy for quality, I use the weekly share of merges that are labeled as bugfixes. The bugfix rate is a proxy for the share of work spent fixing software issues across a codebase. An advantage of this metric is that it corresponds to issues in a codebase that might not be caught immediately after an initial merge. Agent use may affect the bug fixing behavior of organizations in other ways—for example by shifting the focus of engineers from low-priority bugfix work to new feature work.

Both these measures are short-term proxies for quality. If agent-produced code creates technical debt, it may take longer to measure negative effects on quality. If agent-produced code is more standardized and easier to maintain, it may take longer to measure positive effects on quality. Furthermore, these proxies do not measure other quality-relevant patterns, including duplicated logic across the codebase and code complexity.

Estimated effects of the agent feature To quantify effects of the agent feature release, I estimate dynamic difference-in-differences regressions following the specification

$$Y_{i,t} = \alpha_i + \alpha_t + \sum_{k \neq -1} \beta_k \times D_{i,t,k} + \varepsilon_{i,t} \tag{3}$$

across organizations i and calendar weeks t. $D_{i,t,k} = 1$ if organization i is in the eligible group and week t is k weeks relative to the initial, beta release week. α_i and α_t are organization and week fixed effects. This difference-in-differences design can adjust for time trends that unconditionally affect software-producing organizations—for example, lower output on holidays.

Figure 10 reports results from these regressions across the main outcome variable for output. In the Appendix, Table A19 reports pre-period balance statistics, and Figure A20 plots weekly mean values across periods and groups.

To quantify average effects, I compare changes in outcomes between the eligible and baseline organizations. I use the difference-in-differences specification

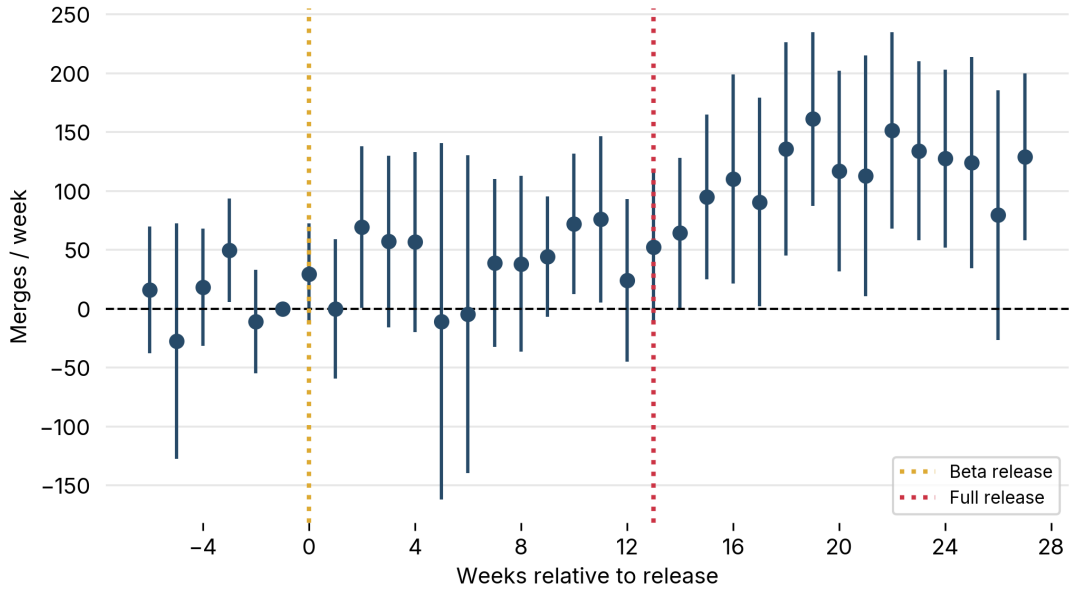
$$Y_{i,t} = \alpha_i + \alpha_t + \beta \times \text{Eligible}_i \times \text{Post}_t + \varepsilon_{i,t} \tag{4}$$

In one set of specifications, I compare outcomes on and after the beta release week to outcomes before the beta release week. In another set of specifications, I compare outcomes on and after the full release week to outcomes before the beta release week, filtering out the intermediate period.

Table 11 reports the results of regressions following Equation (4) across the four outcomes. Over the period after the beta release, Table 11a indicates that merges in the eligible group were 26% higher and lines edited were 29% higher relative to the pre-initial release period.[5] Over the period after the full release, Table 11b indicates that merges in the eligible group were 39% higher and lines edited were 49% higher relative

[5] In the eligible group, the pre-initial release sample mean for merges is 271.7 and for thousands of lines edited is 50.5.

Figure 10: Estimated effects of the agent feature on weekly merges.



Notes – This figure plots dynamic difference-in-differences estimates of effects on merges among organizations eligible for the agent feature. The estimates follow the specification in Equation (3). Vertical lines denote the weeks of the initial release date and full release date of the agent feature. Standard errors are clustered by organization.

to the pre-beta release period. Table 11 also shows no significant changes in revert rates and decreases in bugfix rates. Table A22 reports the results of additional inference using the wild cluster bootstrap-t. Table A23 reports results from additional regressions with logs of output metrics as dependent variables.

Merges and productivity: Scope The results from Figure 10 and Table 11 indicate that merge output increased after the agent was released and made the default generation mode. Can these changes in merge output be interpreted as increases in productivity?

An alternative explanation of these results is that agents cause developers to produce smaller-scoped changes in each merge. Such an explanation would also be consistent with a higher merge rate. However, if this alternative explanation caused an increase in merges, there would be additional implications on merge characteristics that do not appear consistent with the data.

Table A26a reports estimates of the effects of the agent release on proxies for merge scope: Lines edited per merge and files touched per merge. The analysis finds no significant changes in these scope proxies over the sample period. Confidence intervals

Table 11: Agents increase output.

(a) Post beta versus pre beta.

	Merges (1)	Lines Edited (K) (2)	Revert Rate (%) (3)	Bugfix Rate (%) (4)
Eligible × Post Beta Release	70.09***	14.80***	-0.07	-1.41**
	(18.60)	(4.91)	(0.14)	(0.58)
R^2	0.87	0.83	0.27	0.53
Within R^2	0.02	0.01	0.00	0.00
Observations	1,088	1,088	1,088	1,088
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓
Pre-Beta Mean	271.7	50.5	1.07	10.4
Percent Change	25.8%	29.3%	-6.9%	-13.6%

(b) Post full release versus pre beta.

	Merges (1)	Lines Edited (K) (2)	Revert Rate (%) (3)	Bugfix Rate (%) (4)
Eligible × Post Full Release	104.69***	24.61***	-0.08	-2.40***
	(25.54)	(7.48)	(0.17)	(0.70)
R^2	0.90	0.87	0.37	0.66
Within R^2	0.05	0.05	0.00	0.02
Observations	672	672	672	672
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓
Pre-Beta Mean	271.7	50.5	1.07	10.4
Percent Change	38.5%	48.7%	-7.6%	-23.1%

Notes – These tables report difference-in-differences estimates for outcomes related to output (merges and thousands of lines edited) and quality (revert rate and bugfix rate) among organizations eligible for the agent feature. The estimates follow the specification in Equation (4). The top panel compares outcomes from the pre- and post-beta release periods. The bottom panel compares outcomes from the post-full release and pre-beta release periods. Standard errors clustered by organization are reported in parentheses.

rule out changes large enough to offset the increase in merges. Given these results, the alternate explanation would require further assumptions on the effects of agent use:

Agent use would need to have produced verbose code edits that increased lines edited without increasing scope—such edits could keep lines edited per merge consistent while decreasing the scope of each edited line. Even if this further assumption holds, the alternative story would also have to account for no significant decreases in files touched from agent use. Such an assumption would require not only higher agent verbosity over edited lines, but also more frequent agent edits over extraneous files. Developers would have then chosen to merge these extraneous files. According to the evidence on these proxies, merge scope does not appear to have decreased with agent use.

Merges and productivity: Value What kinds of work does agent use affect? Table A26b considers changes in merge output across small and large merges, defined as those below and above the median of lines edited or files touched, with respect to eligible organizations during the pre-release period. I estimate effects on these merge types both in the period between release and default and in the period after default. I find similar effects across merge sizes, which suggests that agent use increases output across merge scopes. If merge scope relates to merge value, these results suggest that the marginal value of merges from agent use is no lower than the value of merges in the pre-release period. However, more comprehensive estimates of value may require longer horizons and organization-specific outcome measures.

Internal and external validity These estimates reflect changes in outcomes for organizations in the sample. How generalizable are these results?

One set of questions relates to internal validity. A challenge to identification would be that a contemporaneous change in some other driver of code merges—which differentially affected the eligible group relative to the baseline group—coincided with the agent's beta release and full release dates. There were no other Cursor client updates in the months of these two dates, which makes it unlikely that another platform event would affect the eligible group around those dates. The platform also did not announce these dates in advance, which makes it unlikely that organizations adjusted merge behavior in anticipation of agent releases. Each organization in the eligible group had already been using the platform for at least six weeks before the agent was announced and released, and the platform did not differentially target feature rollouts according to organization characteristics. Given these institutional details, it is reasonable to assume that the agent's beta and full releases generated exogenous variation in the eligible group's production technology.

Another set of questions relates to external validity. If the early adopters of Cursor who make up the eligible group are experimental organizations that are receptive to new technologies, the estimated effects on this group may be higher than analogous effects of agents on other groups. Such an "experimental organization" characterization is consistent with the eligible group's higher pre-period revert rates according to the balance statistics in Table A19. However a heterogeneity analysis in Table A27 indicates that eligible organizations with higher pre-period revert rates had no larger increases in merges after the agent was released and made default. Moreover, Table A19 indicates that the organizations in the eligible and baseline groups are similar on observables other than the revert rate, including merge frequencies and bugfix rates. The similarity along these balance statistics suggests that other potential confounders, like codebase maintainability, would not contribute to differences across the comparison groups in this sample. Across other dimensions, Table A18 compares characteristics of the eligible and baseline groups to the rest of the sample. Prior to the beta release, the eligible group had comparable numbers of users to the other samples. This suggests similar AI exposure at the time of treatment. Eventually, the number of users in the eligible group increased, and the engineer share fell, which suggests that more workers, and disproportionately non-engineering workers, adopted agents.

5.2. Heterogeneous Increases in Output

If agents benefit verifiable work and expert workers, output effects may differ across firm types. To test these predictions, I extend the difference-in-differences specification by splitting the eligible firms across medians of software engineer share and average work experience of users matched to the firms.

Table 12 reports the results of these regressions. For firms with lower software engineer shares, the increase in weekly merges was 50% after the full release; for firms with higher software engineer shares, the increase was 30%. For firms with lower average work experience, the increase in weekly merges was 23% after the full release; for firms with higher average work experience, the increase was 52%. Table A24 reports analogous regression results using log merges as the dependent variable, and finds similar results.

5.3. Person-Level Evidence on Output

While the main output data is aggregated at the firm level, I use disaggregated data from Company 1 to measure person-level correlates of agent usage. The data covers merges

Table 12: Larger effects for firms with more non-engineers and experienced workers.

(a) Post beta versus pre beta.

	Software Engineer Share		Average Work Experience	
	Low	High	Low	High
Eligible × Post Beta Release	84.51***	55.68**	37.01**	103.18***
	(23.90)	(25.07)	(15.20)	(28.36)
R^2	0.86	0.88	0.88	0.87
Within R^2	0.04	0.01	0.01	0.04
Observations	680	680	680	680
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓
Pre-Beta Mean	232.7	310.8	251.3	292.1
Percent Change	36.3%	17.9%	14.7%	35.3%

(b) Post full release versus pre beta.

	Software Engineer Share		Average Work Experience	
	Low	High	Low	High
Eligible × Post Full Release	116.73***	92.65**	58.11**	151.28***
	(32.14)	(38.56)	(21.08)	(41.25)
R^2	0.89	0.91	0.91	0.91
Within R^2	0.12	0.05	0.03	0.14
Observations	420	420	420	420
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓
Pre-Beta Mean	232.7	310.8	251.3	292.1
Percent Change	50.2%	29.8%	23.1%	51.8%

Notes – These tables report difference-in-differences estimates for weekly merges among organizations eligible for the agent feature. Each pair of regression results splits the eligible group across the median of software engineering share and the median of average work experience. The estimates follow the specification in Equation (4). The top panel compares outcomes from the post-initial release and pre-initial release periods. The bottom panel compares outcomes from the post-full release and pre-initial release periods. Standard errors clustered by organization are reported in parentheses.

Table 13: Correlates of frequent agent usage.

(a) Output and review velocity.

	Total Merges (1)	Total Lines (K) (2)	Lines (K) / Merge (3)	Review Time (h) (4)
Agent Usage (σ)	5.13***	2.94***	0.04**	1.75
	(0.96)	(0.44)	(0.02)	(1.52)
Constant	30.87***	9.79***	0.39***	43.96***
	(0.90)	(0.44)	(0.02)	(1.66)
R^2	0.04	0.05	0.01	0.00
Observations	783	783	783	783

(b) Quality and scope.

	Rollback Rate (%) (5)	Files / Merge (6)	Filetypes / Merge (7)	Areas / Merge (8)
Agent Usage (σ)	-0.06	0.61***	0.08***	0.01
	(0.21)	(0.20)	(0.02)	(0.02)
Constant	3.61***	6.26***	1.68***	1.19***
	(0.26)	(0.19)	(0.02)	(0.02)
R^2	0.00	0.01	0.02	0.00
Observations	783	783	783	783

Notes – These tables report the results of cross-sectional regressions of outcomes on agent usage frequency at Company 1. The estimates follow the specification in Equation (5). One standard deviation in usage corresponds to 459 messages over the two-month period. The top panel reports results on outcomes corresponding to throughput and velocity. Columns (1) and (2) correspond to total outcomes per person, and Columns (3) and (4) correspond to merge-level means per person. The bottom panel reports results on outcomes corresponding to quality and scope. Column (5) corresponds to a rate per person, and columns (6) to (8) correspond to merge-level means per person. Robust standard errors are reported in parentheses.

and deployment outcomes from May and June 2025. Merges and lines edited have the same definitions as in Section 5.1. I also use review time, defined as the hours between a merge request's creation and the final codebase merge. Rollback rate corresponds to the share of merges by a user that the organization attributed to any rollback in deployment. As proxies for scope, I analyze unique files, unique filetypes, and unique codebase areas edited per merge.

To estimate cross-sectional relationships between agent usage frequency and out-

comes, I use the specification

$$Y_i = \alpha + \beta \text{Usage}_i + \varepsilon_i \tag{5}$$

Table 13 reports the results of these regressions.

Higher agent usage is associated with higher output. One standard deviation higher agent use is associated with 5.13 more merges and 2,940 more lines edited over the two-month period. Moreover, one standard deviation higher agent use is associated with 0.61 more files touched per merge and 0.08 more filetypes touched per merge, which could indicate larger-scoped code changes.

These results reflect cross-sectional evidence and, unlike the results from Section 5.1, are not a product of a natural experiment. Therefore, while the cross-sectional results could reflect consequences of agent use, they could also reflect characteristics of output from workers who are more disposed to use agents. For example, more productive workers may be more willing to use new technologies like agents and may work on larger-scoped code changes. The results provide worker-level evidence that is consistent with the aggregate evidence.

Discussion

Agents change AI from a support tool to a delegation tool. As the cost of implementation falls, production becomes bound by specification and verification. This shift to higher-order production can benefit verifiable work and expert workers. This paper studies agents in software production and finds evidence consistent with these implications. Using data from Cursor, I find that agents have become the most common AI instrument used for software development. Designers and workers at consulting firms are among the most frequent users of agents. Experienced workers ask fewer questions to agents, pursue more alignment by planning with agents, and accept agent output at higher rates. Agents increase firm-level output, especially for firms with more non-engineers and more experienced workers.

How might these results extend to other domains? One view is that software production is unique. Software production contains syntactic regularities that AI agents can match, short-run verifiable outputs that can be useful for post-training, and organization-level integration practices that may build safeguards for agent-generated output. Another view is that software production is early. Workers in software firms may be more willing

to try new technologies before they diffuse across the economy. The feedback loop from coding agent developers who consume their own products may have increased the initial speed of development in this domain. While it is true that code outputs are abundant in pre-training data and cheap to produce for post-training data, increased investment in data collection may improve coverage across other domains. Part of the current training paradigm requires reward models that can evaluate how well an agent transforms user intent into output. When there are syntactic regularities—mathematical formalism, accounting standards, contractual patterns, analysis templates—there is room to train agents that allow workers to outsource execution. It is possible that agents may have similar effects in mathematics, accounting, law, and other domains with syntactic regularities.

Agents add a layer of abstraction to production. This move toward abstract production is an ongoing phenomenon. In software development, production has become more abstract as the programming environment has evolved from bit manipulation to assembly code to high-level programming languages. Across other sectors, production has become more abstract as workers use algorithms to diagnose patients, digital workstations to produce music, and algorithms to trade. The evidence in this paper suggests rising returns to worker skills as knowledge work becomes more abstract.

References

Acemoglu, Daron and David Autor, "Skills, Tasks and Technologies: Implications for Employment and Earnings," in "Handbook of Labor Economics," Vol. 4, Elsevier, 2011, pp. 1043–1171. 5

_ **and Pascual Restrepo**, "Automation and new tasks: How technology displaces and reinstates labor," *Journal of economic perspectives*, 2019, *33* (2), 3–30. 1

_ **, Fredric Kong, and Pascual Restrepo**, "Tasks At Work: Comparative Advantage, Technology and Labor Demand," Technical Report w32872, National Bureau of Economic Research, Cambridge, MA August 2024. 5

Aghion, Philippe and Jean Tirole, "Formal and real authority in organizations," *Journal of political economy*, 1997, *105* (1), 1–29. 1

Albarran, Nik, "What New Data Tells Us About the Rise of Agentic AI in Engineering Workflows," Jellyfish Blog August 2025. Accessed: 2026-02-16. 6

Angelova, Victoria, Will Dobbie, and Crystal S Yang, "Algorithmic recommendations and human discretion," *Review of Economic Studies*, 2025, p. rdaf084. 1, 5

Anthropic, "Anthropic Economic Index: AI's Impact on Software Development," April 2025. 6

Arcolano, Nicholas, "Jellyfish and OpenAI Team up to Measure the Impact of AI Coding Tools," August 2025. 6

Athey, Susan and Scott Stern, "The Impact of Information Technology on Emergency Health Care Outcomes," *The Rand Journal of Economics*, 2002, *33* (3), 399–432. 5

Atkinson, Anthony B and Joseph E Stiglitz, "A new view of technological change," *The Economic Journal*, 1969, *79* (315), 573–578. 1

Autor, David and Neil Thompson, "Expertise," Technical Report w33941, National Bureau of Economic Research, Cambridge, MA June 2025. 6

_ **, Frank Levy, and Richard J. Murnane**, "The Skill Content of Recent Technological Change: An Empirical Exploration," *The Quarterly Journal of Economics*, November 2003, *118* (4), 1279–1333. 1, 5

Bartel, Ann, Casey Ichniowski, and Kathryn Shaw, "How does information technology affect productivity? Plant-level comparisons of product innovation, process improvement, and worker skills," *The quarterly journal of Economics*, 2007, *122* (4), 1721–1758. 1

Becker, Joel, Nate Rush, Elizabeth Barnes, and David Rein, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity," July 2025. 6

Bick, Alexander, Adam Blandin, and David Deming, "The Rapid Adoption of Generative AI," Technical Report w32966, National Bureau of Economic Research, Cambridge, MA September 2024. 7

Bloom, Nicholas, Luis Garicano, Raffaella Sadun, and John Van Reenen, "The Distinct Effects of Information Technology and Communication Technology on Firm Organization," *Management Science*, December 2014, *60* (12), 2859–2885. 5

Bresnahan, Timothy F. and Manuel Trajtenberg, "General Purpose Technologies 'Engines of Growth'?," *Journal of Econometrics*, January 1995, *65* (1), 83–108. 5

Brynjolfsson, Erik and Lorin M Hitt, "Beyond Computation: Information Technology, Organizational Transformation and Business Performance," *Journal of Economic Perspectives*, November 2000, *14* (4), 23–48. 5

_ , Bharat Chandar, and Ruyu Chen, "Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of Artificial Intelligence," Technical Report, Working paper. 2025. 6

_ , Danielle Li, and Lindsey Raymond, "Generative AI at Work," *The Quarterly Journal of Economics*, April 2025, *140* (2), 889–942. 1, 5, 7

Cameron, A Colin, Jonah B Gelbach, and Douglas L Miller, "Bootstrap-Based Improvements for Inference with Clustered Errors," *The review of economics and statistics*, 2008, *90* (3), 414–427. 68

Chatterji, Aaron, Thomas Cunningham, David Deming, Zoe Hitzig, Christopher Ong, Carl Yan Shan, and Kevin Wadman, "How People Use ChatGPT," Technical Report w34255, National Bureau of Economic Research, Cambridge, MA September 2025. 5, 7

Chen, Bei, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen, "CodeT: Code Generation with Generated Tests," November 2022. 10

Chen, Fiona and James Stratton, "Artificial Intelligence in the Firm," January 2026. 6, 15

Chen, Mark, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba, "Evaluating Large Language Models Trained on Code," July 2021. 10

Chen, Valerie, Ameet Talwalkar, Robert Brennan, and Graham Neubig, "Code with me or for me? how increasing ai automation transforms developer workflows," *arXiv preprint arXiv:2507.08149*, 2025. 6

Clark, A. and D. Chalmers, "The Extended Mind," *Analysis*, January 1998, *58* (1), 7–19. 5

Cui, Zheyuan Kevin, Mert Demirer, Sonia Jaffe, Leon Musolff, Sida Peng, and Tobias Salz, "The Effects of Generative Ai on High Skilled Work: Evidence from Three Field Experiments with Software Developers," *Available at SSRN 4945566*, 2024. 6

Daniotti, Simone, Johannes Wachs, Xiangnan Feng, and Frank Neffke, "Who is using AI to code? Global diffusion and impact of generative AI," *Science*, January 2026, *391* (6776), 373–379. 6

David, Paul A, "The dynamo and the computer: an historical perspective on the modern productivity paradox," *The American economic review*, 1990, *80* (2), 355–361. 1

Dell'Acqua, Fabrizio, Edward McFowland III, Ethan R Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Krayer, Francois Candelon, and Karim R Lakhani, "Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality," *Harvard Business School Technology & Operations Mgt. Unit Working Paper*, 2023. 6

Dillon, Eleanor Wiske, Sonia Jaffe, Nicole Immorlica, and Christopher T. Stanton, "Shifting Work Patterns with Generative AI," July 2025. 6

Fischer, Alexander and David Roodman, "Fwildclusterboot: Fast Wild Cluster Bootstrap Inference for Linear Regression Models (Version 0.14.0)," 2021. 68

Forsgren, Nicole, Dustin Smith, Jez Humble, and Jessie Frazelle, "2019 Accelerate State of Devops Report," *DORA and Google Cloud, Tech. Rep*, 2019, *48455*. 10

_ , Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler, "The SPACE of Developer Productivity: There's More to It than You Think.," *Queue*, 2021, *19* (1), 20–48. 10

Fowler, Martin, "Continuous Integration," January 2024. 10

Friedman, Nat, "Introducing GitHub Copilot: Your AI Pair Programmer," June 2021. 1, 5

Garicano, Luis and Esteban Rossi-Hansberg, "Organization and Inequality in a Knowledge Economy," *The Quarterly Journal of Economics*, November 2006, *121* (4), 1383–1435. 6

Gimbel, Martha, Molly Kinder, Joshua Kendall, and Maddie Lee, "Evaluating the Impact of AI on the Labor Market: Current State of Affairs," October 2025. 6

Hadfield, Gillian K. and Andrew Koh, "An Economy of AI Agents," September 2025. 7

Hampole, Menaka, Dimitris Papanikolaou, Lawrence D.W. Schmidt, and Bryan Seegmiller, "Artificial Intelligence and the Labor Market," Technical Report w33509, National Bureau of Economic Research, Cambridge, MA February 2025. 6

Handa, Kunal, Alex Tamkin, Miles McCain, Saffron Huang, Esin Durmus, Sarah Heck, Jared Mueller, Jerry Hong, Stuart Ritchie, Tim Belonax, Kevin K. Troy, Dario Amodei, Jared Kaplan, Jack Clark, and Deep Ganguli, "Which Economic Tasks Are Performed with AI? Evidence from Millions of Claude Conversations," February 2025. 7

He, Hao, Courtney Miller, Shyam Agarwal, Christian Kästner, and Bogdan Vasilescu, "Speed at the Cost of Quality: How Cursor AI Increases Short-Term Velocity and Long-Term Complexity in Open-Source Projects," *arXiv preprint arXiv:2511.04427*, November 2025. 6

Hindle, Abram, Earl T. Barr, Mark Gabel, Zhendong Su, and Premkumar Devanbu, "On the Naturalness of Software," *Communications of the ACM*, April 2016, *59* (5), 122–131. 10

Hoffmann, Manuel, Sam Boysel, Frank Nagle, Sida Peng, and Kevin Xu, "Generative AI and the Nature of Work," 2024. 6, 15

Holmström, Bengt, "Moral hazard and observability," *The Bell journal of economics*, 1979, pp. 74–91. 1

Humlum, Anders and Emilie Vestergaard, "Large Language Models, Small Labor Market Effects," Technical Report w33777, National Bureau of Economic Research, Cambridge, MA May 2025. 5, 7

Hutchins, Edwin, *Cognition in the Wild* A Bradford Book, 8. pr ed., Cambridge, Mass.: MIT Press, 2006. 5

Ide, Enrique and Eduard Talamàs, "Artificial Intelligence in the Knowledge Economy," *Journal of Political Economy*, October 2025, pp. 000–000. 1, 4, 6

Imas, Alex, Kevin Lee, and Sanjog Misra, "Agentic Interactions," *SSRN Electronic Journal*, December 2025, (5875162). SSRN working paper. 7

Katz, Lawrence F. and Kevin M. Murphy, "Changes in Relative Wages, 1963-1987: Supply and Demand Factors," *The Quarterly Journal of Economics*, February 1992, *107* (1), 35–78. 1, 6

Kirsh, David and Paul Maglio, "On Distinguishing Epistemic from Pragmatic Action," *Cognitive Science*, October 1994, *18* (4), 513–549. 5

Korinek, Anton, "Generative AI for Economic Research: Use Cases and Implications for Economists," *Journal of Economic Literature*, December 2023, *61* (4), 1281–1317. 7

Le, Hung, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi, "CodeRL: Mastering Code Generation through Pretrained Models and Deep Reinforcement Learning," November 2022. 10

Liang, Annie, "Artificial Intelligence Clones," April 2025. 7

Lichtinger, Guy and Seyed Mahdi Hosseini Maasoum, "Generative AI as Seniority-Biased Technological Change: Evidence from U.S. Résumé and Job Posting Data," 2025. 6

Mansfield, Edwin, "Technical Change and the Rate of Imitation," *Econometrica*, October 1961, *29* (4), 741. 5

Martin, Robert C., *Clean Code: A Handbook of Agile Software Craftsmanship* Robert C. Martin Series, Prentice Hall, 2012. 10

McElheran, Kristina, J. Frank Li, Erik Brynjolfsson, Zachary Kroff, Emin Dinlersoz, Lucia Foster, and Nikolas Zolas, "AI Adoption in America: Who, What, and Where," *Journal of Economics & Management Strategy*, March 2024, *33* (2), 375–415. 7

Mozannar, Hussein, Gagan Bansal, Cheng Tan, Adam Fourney, Victor Dibia, Jingya Chen, Jack Gerrits, Tyler Payne, Matheus Kunzler Maldaner, Madeleine Grunde-McLaughlin et al., "Magentic-ui: Towards human-in-the-loop agentic systems," *arXiv preprint arXiv:2507.22358*, 2025. 3, 14

Noy, Shakked and Whitney Zhang, "Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence," *Science*, July 2023, *381* (6654), 187–192. 5, 7

Paradis, Elise, Kate Grey, Quinn Madison, Daye Nam, Andrew Macvean, Vahid Meimand, Nan Zhang, Ben Ferrari-Church, and Satish Chandra, "How Much Does AI Impact Development Speed? An Enterprise-Based Randomized Controlled Trial," November 2024. 6

Peng, Sida, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer, "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot," February 2023. 6

Risko, Evan F and Sam J Gilbert, "Cognitive Offloading," *Trends in cognitive sciences*, 2016, *20* (9), 676–688. 5

Rothschild, David M., Markus Mobius, Jake M. Hofman, Eleanor W. Dillon, Daniel G. Goldstein, Nicole Immorlica, Sonia Jaffe, Brendan Lucier, Aleksandrs Slivkins, and Matthew Vogel, "The Agentic Economy," May 2025. 7

Sadowski, Caitlin, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli, "Modern Code Review: A Case Study at Google," in "Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice" ACM Gothenburg Sweden May 2018, pp. 181–190. 10

Schick, Timo, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom, "Toolformer: Language Models Can Teach Themselves to Use Tools," in A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds., *Advances in Neural Information Processing Systems*, Vol. 36 Curran Associates, Inc. 2023, pp. 68539–68551. 1

Shahidi, Peyman, Gili Rusak, Benjamin S Manning, Andrey Fradkin, and John J Horton, "The Coasean Singularity? Demand, Supply, and Market Design with AI Agents," *NBER Chapters*, 2025. 7

Shao, Yijia, Humishka Zope, Yucheng Jiang, Jiaxin Pei, David Nguyen, Erik Brynjolfsson, and Diyi Yang, "Future of Work with AI Agents: Auditing Automation and Augmentation Potential across the U.S. Workforce," June 2025. 7

Shen, Judy Hanwen and Alex Tamkin, "How AI Impacts Skill Formation," *arXiv preprint arXiv:2601.20245*, January 2026. 6

Simon, Herbert A, "The structure of ill structured problems," *Artificial intelligence*, 1973, *4* (3-4), 181–201. 1

Syverson, Chad, "What Determines Productivity?," *Journal of Economic Literature*, June 2011, *49* (2), 326–365. 5

Thompson, Clive, "Coding After Coders: The End of Computer Programming as We Know It," *The New York Times Magazine*, March 2026. 7

Tomlinson, Kiran, Sonia Jaffe, Will Wang, Scott Counts, and Siddharth Suri, "Working with AI: Measuring the Applicability of Generative AI to Occupations," September 2025. 7

Vafa, Keyon, Ashesh Rambachan, and Sendhil Mullainathan, "Do Large Language Models Perform the Way People Expect? Measuring the Human Generalization Function," in "International Conference on Machine Learning" PMLR 2024, pp. 48919–48937. 6

Weidmann, Ben, Yixian Xu, and David J Deming, "Measuring human leadership skills with AI agents," Technical Report, National Bureau of Economic Research 2025. 6

Yao, Shunyu, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao, "React: Synergizing Reasoning and Acting in Language Models," in "International Conference on Learning Representations (ICLR)" 2023. 1

Zhang, Jiajie and Donald A. Norman, "Representations in Distributed Cognitive Tasks," *Cognitive Science*, January 1994, *18* (1), 87–122. 5

Zhao, Wenting, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng, "WildChat: 1M ChatGPT Interaction Logs in the Wild," May 2024. 7

A. Appendix

Section A.1 includes derivations for the organizing framework. Section A.2 includes details on labeling work history and message intent. Section A.3 includes additional analysis related to agent usage. Section A.4 includes additional analysis related to merge output.

A.1. Additional Derivations

Define the delegation objective

$$\mathcal{L}(s; v_j, e_i) = \frac{s^2}{2(1+\lambda e_i)} + \frac{\bar{\tau}(1-v_j)}{1+\mu e_i} + \exp\{-(\beta_s s + \beta_v v_j)\}R,$$

so that $C^A(v_j, e_i) = \min_{s \geq 0} \mathcal{L}(s; v_j, e_i)$.

Existence and interior solution For fixed (v_j, e_i), $\mathcal{L}$ is strictly convex in s and coercive, so there is a unique minimizer s^*. The derivative with respect to s is

$$\frac{\partial \mathcal{L}}{\partial s} = \frac{s}{1+\lambda e_i} - \beta_s \exp\{-(\beta_s s + \beta_v v_j)\}R.$$

Because $\beta_s > 0$ and $R > 0$,

$$\left.\frac{\partial \mathcal{L}}{\partial s}\right|_{s=0} = -\beta_s \exp\{-\beta_v v_j\}R < 0,$$

and $\partial \mathcal{L}/\partial s \to \infty$ as $s \to \infty$, implying an interior optimum. This interiority result is used in the main text and in Predictions 2–4 below.

Prediction 1 For any fixed s, the objective is strictly decreasing in v_j:

$$\frac{\partial \mathcal{L}}{\partial v_j} = -\frac{\bar{\tau}}{1+\mu e_i} - \beta_v \exp\{-(\beta_s s + \beta_v v_j)\}R < 0.$$

Therefore, $\mathcal{L}(s; v_j, e_i)$ is pointwise strictly decreasing in v_j, so its minimum $C^A(v_j, e_i)$ is strictly decreasing in v_j.

Prediction 2 Using the interiority result above, the FOC is

$$\frac{s^*}{1+\lambda e_i} = \beta_s R \exp\{-(\beta_s s^* + \beta_v v_j)\}.$$

Implicit differentiation yields

$$\frac{\partial s^*}{\partial e_i} = \frac{\frac{\lambda s^*}{(1+\lambda e_i)^2}}{\frac{1}{1+\lambda e_i} + \beta_s^2 R \exp\{-(\beta_s s^* + \beta_v v_j)\}} > 0.$$

The numerator and denominator are strictly positive, so $\partial s^*/\partial e_i > 0$.

Prediction 3 Under delegation,

$$\frac{d}{de_i} \log p(s^*, v_j) = -\beta_s \frac{\partial s^*}{\partial e_i}.$$

By Prediction 2 above, $\partial s^*/\partial e_i > 0$, so

$$\frac{d}{de_i} \log p(s^*, v_j) < 0.$$

Therefore, misalignment falls with expertise.

Prediction 4 By the envelope theorem,

$$-\frac{\partial C^A}{\partial e_i} = \frac{\lambda (s^*)^2}{2(1+\lambda e_i)^2} + \frac{\mu \bar{\tau}(1-v_j)}{(1+\mu e_i)^2}.$$

For copilot use,

$$-\frac{\partial C^C}{\partial e_i} = \frac{(1-\alpha)\phi}{(1+\phi e_i)^2} - \kappa'(e_i).$$

Therefore, delegation has a strictly larger return to expertise if and only if

$$\frac{\lambda (s^*)^2}{2(1+\lambda e_i)^2} + \frac{\mu \bar{\tau}(1-v_j)}{(1+\mu e_i)^2} > \frac{(1-\alpha)\phi}{(1+\phi e_i)^2} - \kappa'(e_i),$$

A.2. Labeling Details

User characteristics Role, seniority, and years of work experience of sampled users were determined by a labeling prompt combined with in-context examples. Sector is an internal Cursor designation at the organization level. De-identified data on work history summaries was constructed by Cursor by matching a subset of users to publicly available profile information. Each de-identified history summary used for labeling contained a list of job titles and year ranges. The 100 most common job titles were manually labeled with role and seniority, and passed as examples alongside a prompt detailed in Figure A1 to label the history summaries. The labeling was conducted using an internal inference endpoint.

Figure A1: Characteristics prompt.

User Characteristics From Work History

Given a JSON array of work history, extract the following fields:

- earliest_professional_employment_start_year: the chronologically first professional employment start year as a 4-digit integer; ignore any student/intern/volunteer/teaching/campus/research assistant roles.
- role: choose ONE from this set exactly: {role_options}.
- seniority: choose ONE from this set exactly: {seniority_options}.

Classification guidelines:

- Weigh the current/most recent title most strongly when determining role.
- If the role is general management, people management, operations, or founder, and does not clearly map to software, product, design, data, marketing, or sales management, set role to 'business/operations'.

Seniority guidelines:

- Set seniority='executive/founder' ONLY for explicit executive signals: C-level (e.g., CEO, CTO, CFO, COO, CIO, CISO ...), President, Vice President/VP/SVP/EVP, Founder/Co-Founder, or titles containing 'Global Head' or 'Worldwide Head'.
- Otherwise, classify titles containing 'Director', 'Senior Director', 'Executive Director', 'Head of', or 'Manager' as seniority='management' (not 'executive/founder').
- management: People management, including Director, Senior Director, Executive Director, Head of, Manager, Senior Manager, Principal Manager, Group Manager (not 'executive/founder').
- staff/principal: Staff, Principal, Distinguished, Fellow, Architect, Solutions/Software Architect. Numeric levels IV/V (or 4/5) map here when IC roles.
- senior: Senior, Lead (when not 'Team Lead' and not paired with Manager/Director/Head), IC level III (or 3).
- junior: Software Engineer, Product Manager, Associate, Junior, IC level I/II (or 1/2). Default for plain 'Engineer/Developer' without modifiers.
- Conflicts: management overrides senior/staff terms in the same title (e.g., 'Senior Director' => management).

Examples of labeled data (use these as guidance):
{examples_text}
Return ONLY valid JSON with keys: earliest_professional_employment_start_year, role, seniority. Do not include comments or markdown fences.

Table A2: Summary statistics on user characteristics.

	Full Sample (1)	Telemetry Snapshot (2)	Agent Metrics (3)	Tab Metrics (4)	Intent Metrics (5)
N users	119,960	82,332	78,368	62,598	42,348
N organizations	1,000	997	994	995	399
Role:					
Software	88,443 (73.7%)	63,507 (77.1%)	60,572 (77.3%)	50,165 (80.1%)	33,563 (79.3%)
Data/ML	10,803 (9.0%)	7,458 (9.1%)	6,997 (8.9%)	5,961 (9.5%)	3,712 (8.8%)
Business/Operations	10,008 (8.3%)	5,611 (6.8%)	5,328 (6.8%)	3,406 (5.4%)	2,622 (6.2%)
Product	5,142 (4.3%)	2,764 (3.4%)	2,603 (3.3%)	1,401 (2.2%)	1,062 (2.5%)
Design	2,884 (2.4%)	1,637 (2.0%)	1,576 (2.0%)	853 (1.4%)	763 (1.8%)
Sales	1,480 (1.2%)	742 (0.9%)	710 (0.9%)	467 (0.7%)	334 (0.8%)
Marketing	1,200 (1.0%)	613 (0.7%)	582 (0.7%)	345 (0.6%)	292 (0.7%)
Sector:					
Software & Developer Tools	48,327 (40.3%)	33,683 (40.9%)	31,900 (40.7%)	26,058 (41.6%)	16,327 (38.6%)
Finance & Fintech	24,875 (20.7%)	16,761 (20.4%)	15,901 (20.3%)	12,827 (20.5%)	9,641 (22.8%)
Consumer & Retail	14,434 (12.0%)	9,825 (11.9%)	9,341 (11.9%)	6,882 (11.0%)	5,571 (13.2%)
Media & Advertising	10,870 (9.1%)	7,623 (9.3%)	7,310 (9.3%)	5,844 (9.3%)	3,831 (9.0%)
Logistics & Platforms	8,973 (7.5%)	6,113 (7.4%)	5,895 (7.5%)	4,738 (7.6%)	3,477 (8.2%)
Healthcare & Life Sciences	7,081 (5.9%)	4,830 (5.9%)	4,647 (5.9%)	3,641 (5.8%)	2,312 (5.5%)
Consulting & Professional Services	5,400 (4.5%)	3,497 (4.2%)	3,374 (4.3%)	2,608 (4.2%)	1,189 (2.8%)
Seniority:					
Junior	46,309 (38.6%)	33,676 (40.9%)	32,312 (41.2%)	26,801 (42.8%)	17,901 (42.3%)
Senior	36,100 (30.1%)	25,523 (31.0%)	24,381 (31.1%)	19,604 (31.3%)	13,527 (31.9%)
Management	19,724 (16.4%)	11,521 (14.0%)	10,747 (13.7%)	7,364 (11.8%)	4,978 (11.8%)
Staff/Principal	13,252 (11.0%)	9,045 (11.0%)	8,507 (10.9%)	7,048 (11.3%)	4,914 (11.6%)
Executive/Founder	4,575 (3.8%)	2,567 (3.1%)	2,421 (3.1%)	1,781 (2.8%)	1,028 (2.4%)
Years of experience:					
Mean	12.1	11.6	11.6	11.3	11.4
Standard deviation	7.0	6.8	6.8	6.7	6.7
25th percentile	7	7	6	6	6
Median	11	10	10	10	10
75th percentile	16	15	15	15	15

Notes – This table reports summary statistics of user characteristics. Column (1) reports statistics from the full sample of organization users. Column (2) reports summary statistics from the telemetry snapshot subsample between the week of November 17, 2025 through the week of January 19, 2026. Column (3) restricts the snapshot subsample to agent users only, and Column (4) restricts the snapshot subsample to autocomplete users only. Column (5) restricts to a subsample of the agent metrics users for whom message intent was collected.

A.3. Additional Exhibits on Agent Usage

I include additional exhibits related to the analysis of agent usage.

- Table A3 reports differences in mean usage between the full agent metrics sample and the message intent subsample.
- Figure A4 reports AI tool usage share across joining cohorts.
- Figure A5 reports the average share of task categories across worker characteristics.
- Figure A6 reports the share of agent-only usage across worker characteristics.
- Figure A7 and Figure A8 report intent shares for all messages and first messages across worker characteristics.
- Table A9 and Table A10 report relationships between first message intents and years of experience, and Table A11 and Table A12 report relationships between all message intents and years of experience.
- Figure A13 reports agent accept rates across worker characteristics.
- Figure A14 reports autocomplete accept rates across worker characteristics.
- Table A15 reports regressions of agent accept rates on work experience in the message intent subsample.
- Table A16 reports relationships between agent accept rates, work experience, plan rates, and ask rates.
- Table A17 reports relationships between autocomplete accept rates and work experience.

Usage statistics Table A3 reports mean usage across samples. Organizations that use the conversation classification feature tend to have higher average usage.

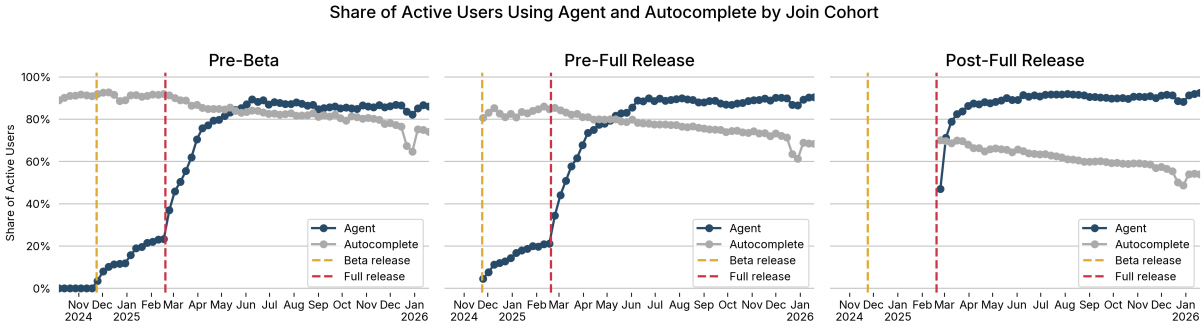
Table A3: Usage statistics across the metrics samples.

	Agent Metrics (1)	Agent Metrics (Intent Window) (2)	Agent Metrics (Intent Match) (3)
N users	78,368	67,838	42,348
N organizations	994	989	399
Mean usage	59.2	60.7	67.6
Role:			
Software	58.8	60.1	66.2
Data/ML	56.0	57.8	65.8
Business/Operations	63.2	65.0	75.3
Product	62.1	64.6	81.1
Design	80.6	83.0	101.8
Sales	56.7	59.8	71.8
Marketing	59.2	66.5	74.3
Sector:			
Software & Developer Tools	61.2	63.6	70.8
Finance & Fintech	57.0	57.4	63.8
Consumer & Retail	59.3	60.0	69.4
Media & Advertising	57.0	58.5	61.7
Logistics & Platforms	57.2	59.2	66.0
Healthcare & Life Sciences	56.3	55.9	63.8
Consulting & Professional Services	63.7	63.8	76.2
Seniority:			
Junior	59.9	60.9	67.5
Senior	57.0	57.9	63.7
Management	56.4	59.7	71.4
Staff/Principal	62.9	65.1	71.8
Executive/Founder	71.7	76.6	85.4
Years of experience quintile:			
1 (least)	59.8	59.8	67.2
2	57.7	59.3	65.4
3	56.8	58.1	64.1
4	59.0	61.2	68.4
5 (most)	63.3	65.8	74.2

Notes – This table reports statistics on mean usage across user characteristics. Column (1) reports statistics from the full agent metrics snapshot sample between the weeks of November 17, 2025 and January 19, 2026. Column (2) restricts to the subsample between the weeks of December 22, 2025 and January 19, 2026, when the message intent data was collected. Column (3) further restricts to the sample matched to message intent.

Agent diffusion across cohorts Figure A4 plots AI tool usage across time, separated by user creation cohort. Later adopters who joined after the full release are more than 30 percentage points more likely to use agent than autocomplete. Early adopters have also shifted to using agents.

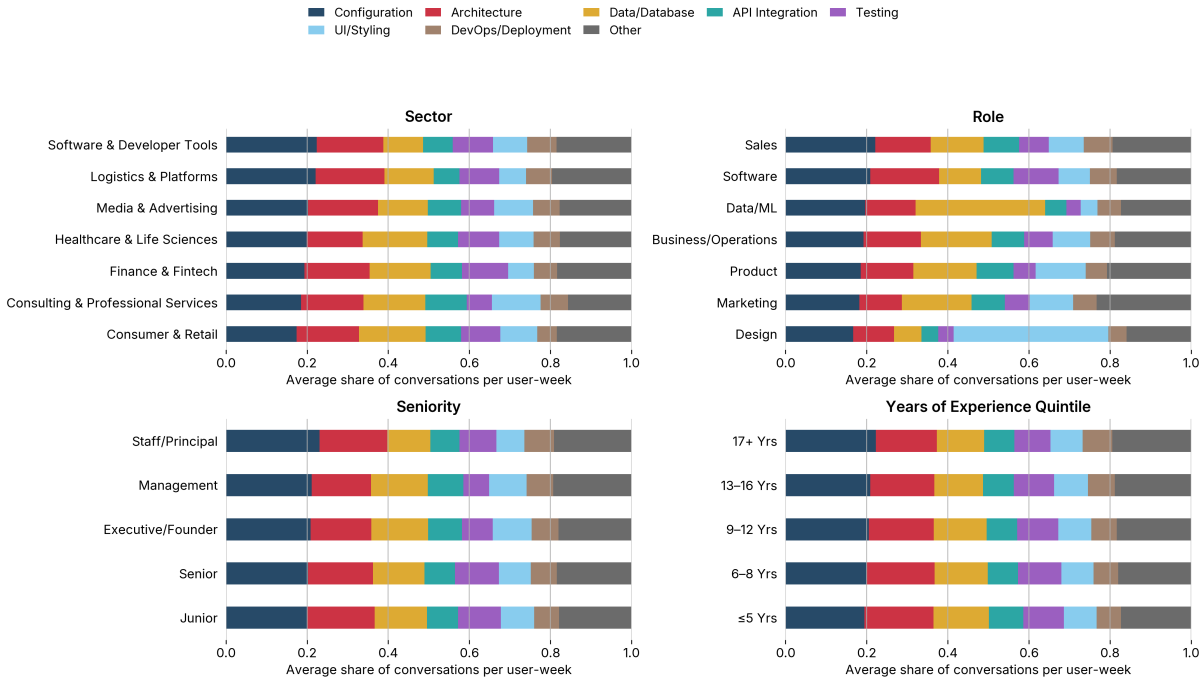
Figure A4: AI tool usage among active users by creation date cohort.



Notes – This figure plots the share of weekly active users who used the autocomplete feature or the agent feature. The agent feature was released in the November 24, 2024 update, and was made the default generation mode in the February 19, 2025 update. The agent would become the default setting when users updated their software clients. Each subfigure corresponds to a different user creation range. Pre-Beta corresponds to users created before November 24, 2024, Pre-Full Release corresponds to users created starting November 24, 2024 and before February 19, 2025, and Post-Full Release corresponds to users created starting February 19, 2025.

Task categories across characteristics Figure A5 plots shares of conversation task categories, averaged across user-weeks.

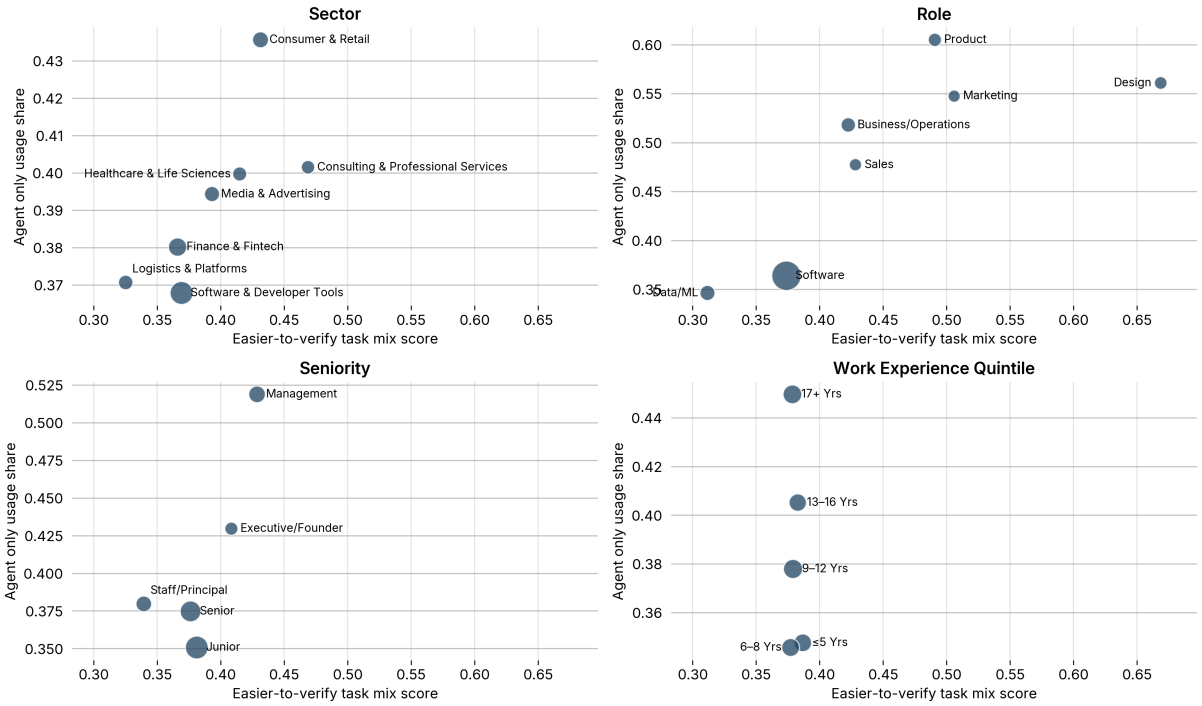
Figure A5: Task categories across characteristics.



Notes – This figure plots the distribution of conversation task category classifications, averaged across user-weeks, by worker characteristics.

Agent-only usage by verifiability score Figure A6 plots agent-only usage shares across worker characteristic-level verifiability scores.

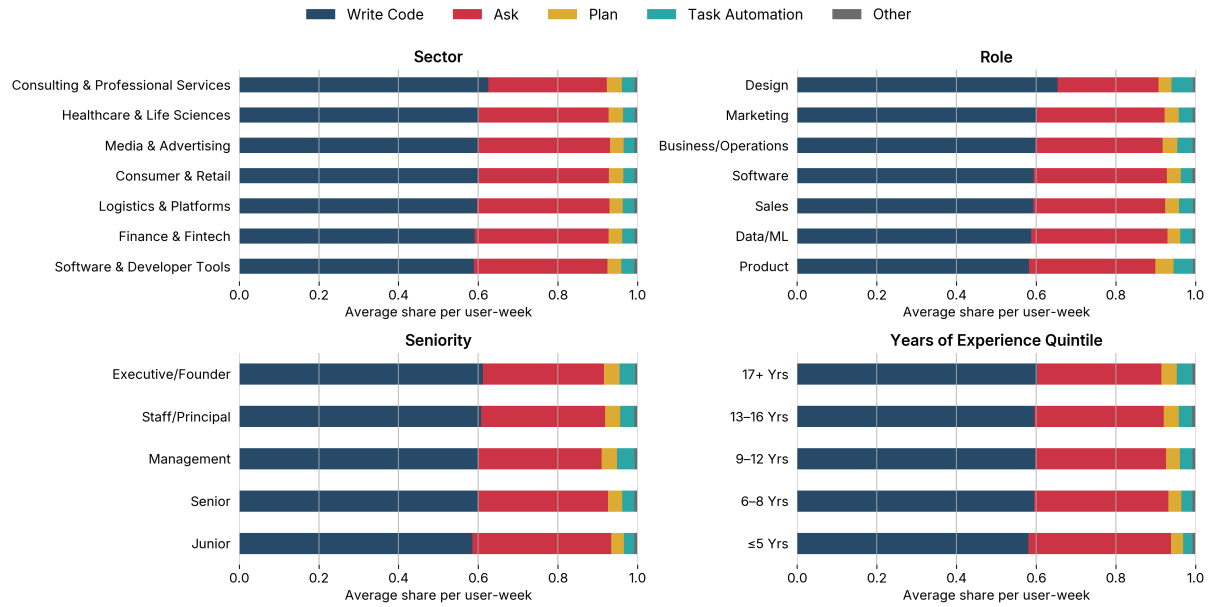
Figure A6: Agent-only usage by verifiability score.



Notes – This figure plots the share of active user-weeks that only used agents across characteristics, sorted by task verifiability score.

Message intents by characteristics Figure A7 plots message intents across worker characteristics.

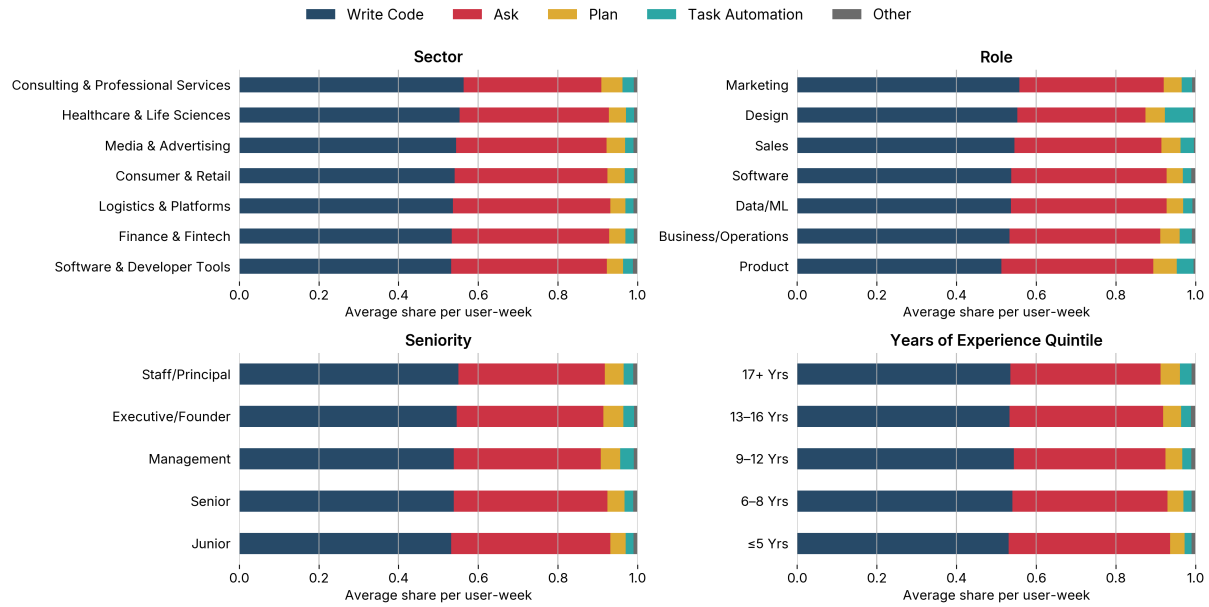
Figure A7: Message intents by worker characteristics.



Notes – This figure plots the average shares of message intents across sectors, roles, seniorities, and quintiles of work experience.

First message intents by characteristics Figure A8 plots message intents across worker characteristics.

Figure A8: First message intents by worker characteristics.



Notes – This figure plots the average shares of first message intents across sectors, roles, seniorities, and quintiles of work experience.

First message intents and years of experience Table A9 and Table A10 report the results of regressions of first message intent rates on years of work experience.

Table A9: First message intents and work experience.

	First Message Intent Share [%]				
	Write Code	Ask	Plan	Task Automation	Other
Work Experience (σ)	0.04	-0.88***	0.45***	0.39***	-0.00
	(0.13)	(0.13)	(0.05)	(0.04)	(0.02)
Constant	53.71***	38.77***	4.21***	2.28***	1.03***
	(0.13)	(0.12)	(0.04)	(0.03)	(0.02)
R^2	0.000	0.001	0.001	0.002	0.000
Observations	118,407	118,407	118,407	118,407	118,407

Notes – This table reports results of regressions of first message intent rates on work experience. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

Table A10: First message intents and work experience.

	First Message Intent Share [%]				
	Write Code	Ask	Plan	Task Automation	Other
Work Experience (σ)	0.12	-0.85***	0.42***	0.31***	0.00
	(0.14)	(0.13)	(0.05)	(0.04)	(0.02)
R^2	0.004	0.005	0.003	0.007	0.001
Within R^2	0.000	0.001	0.001	0.001	0.000
Observations	118,407	118,407	118,407	118,407	118,407
Week FE	✓	✓	✓	✓	✓
Role FE	✓	✓	✓	✓	✓
Sector FE	✓	✓	✓	✓	✓

Notes – This table reports results of regressions of first message intent rates on work experience. The specifications include week, role, and sector fixed effects. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

Message intents and years of experience Table A11 and Table A12 report the results of regressions of message intent rates on years of work experience.

Table A11: Message intents and work experience.

	Message Intent Share [%]				
	Write Code	Ask	Plan	Task Automation	Other
Work Experience (σ)	0.64***	-1.46***	0.28***	0.54***	0.00
	(0.10)	(0.09)	(0.02)	(0.03)	(0.01)
Constant	59.41***	33.24***	3.48***	3.08***	0.79***
	(0.10)	(0.09)	(0.02)	(0.03)	(0.01)
R^2	0.001	0.004	0.002	0.007	0.000
Observations	118,407	118,407	118,407	118,407	118,407

Notes – This table reports results of regressions of message intent rates on work experience. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

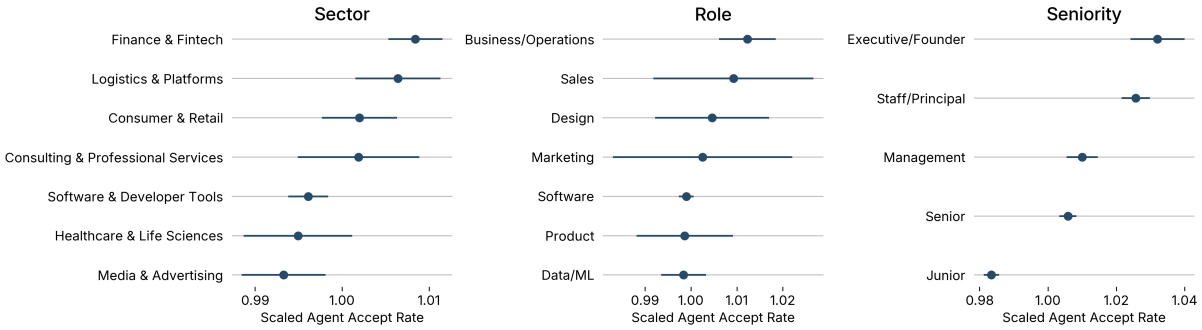
Table A12: Message intents and work experience.

	Message Intent Share [%]				
	Write Code	Ask	Plan	Task Automation	Other
Work Experience (σ)	0.65***	-1.40***	0.26***	0.49***	0.01
	(0.10)	(0.09)	(0.03)	(0.03)	(0.01)
R^2	0.008	0.011	0.006	0.011	0.001
Within R^2	0.001	0.004	0.002	0.005	0.000
Observations	118,407	118,407	118,407	118,407	118,407
Week FE	✓	✓	✓	✓	✓
Role FE	✓	✓	✓	✓	✓
Sector FE	✓	✓	✓	✓	✓

Notes – This table reports results of regressions of message intent rates on work experience. The specifications include week, role, and sector fixed effects. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

Agent accept rates Figure A13 plots agent accept rates across user characteristics.

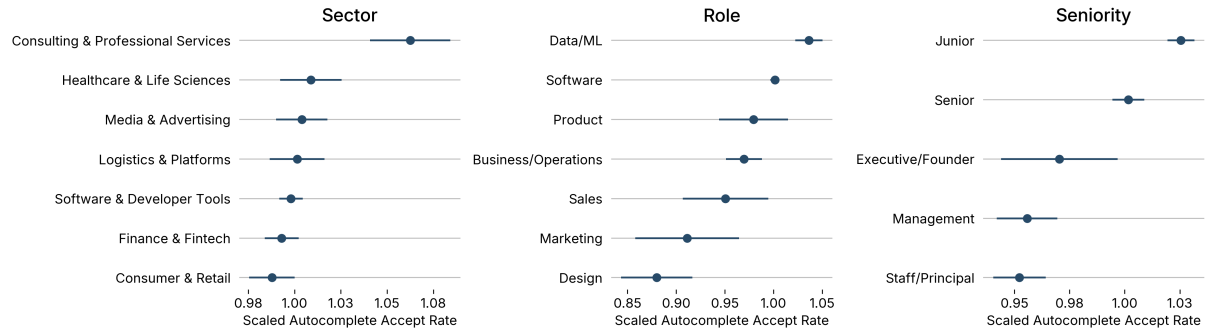
Figure A13: Relative agent accept rates by user characteristics.



Notes – This figure plots relative agent accept rates across sectors, roles, and seniorities.

Autocomplete accept rates Figure A14 plots autocomplete accept rates across user characteristics.

Figure A14: Relative autocomplete accept rates by user characteristics.



Notes – This figure plots relative autocomplete accept rates across sectors, roles, and seniorities.

Accept rates and work experience Table A15 reports the relationship between agent accept rates and work experience, restricting to the set of user-weeks with available message intent data.

Table A15: Accept rates and work experience (message intent sample).

	Relative Accept Rate × 100				
	(1)	(2)	(3)	(4)	(5)
Work Experience (σ)	1.93***	1.92***	1.93***	1.96***	1.65***
	(0.08)	(0.08)	(0.08)	(0.09)	(0.10)
Constant	100.95***				
	(0.09)				
R^2	0.006	0.007	0.007	0.007	0.008
Within R^2	-	0.006	0.006	0.006	0.003
Observations	118,407	118,407	118,407	118,407	118,407
Week FE		✓	✓	✓	✓
Role FE			✓	✓	✓
Sector FE				✓	✓
Seniority FE					✓

Notes – This table reports results of regressions of relative agent accept rates on work experience. The sample is restricted to the set of user-weeks with available message intent data. The columns progressively add week, role, sector, and seniority fixed effects. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

Accept rates, experience, and intents Table A16 reports the relationship between agent accept rates, work experience, and first message intent rates. Experienced workers, users who plan more, and users who ask less accept agent-generated code at higher rates.

Table A16: Correlates of accept rates.

	Relative Accept Rate × 100			
	(1)	(2)	(3)	(4)
Work Experience (σ)	1.65***			1.61***
	(0.10)			(0.09)
Plan Rate [%]		0.07***		0.06***
		(0.01)		(0.01)
Ask Rate [%]			-0.04***	-0.04***
			(0.00)	(0.00)
R^2	0.008	0.006	0.008	0.013
Within R^2	0.003	0.001	0.004	0.008
Observations	118,407	118,407	118,407	118,407
Week FE	✓	✓	✓	✓
Role FE	✓	✓	✓	✓
Sector FE	✓	✓	✓	✓
Seniority FE	✓	✓	✓	✓

Notes – This table reports results of regressions of relative agent accept rates on work experience, ask rates, and plan rates. The specifications include week, role, sector, and seniority fixed effects. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

Autocompletion accept rates and work experience Table A17 reports the relationship between autocomplete accept rates and work experience.

Table A17: Work experience and autocompletion accept rates.

	Relative Accept Rate × 100				
	(1)	(2)	(3)	(4)	(5)
Work Experience (σ)	-4.18***	-4.13***	-4.06***	-4.05***	-3.45***
	(0.18)	(0.18)	(0.18)	(0.18)	(0.20)
Constant	100.00***				
	(0.17)				
R^2	0.005	0.006	0.007	0.007	0.008
Within R^2	-	0.005	0.005	0.005	0.003
Observations	350,362	350,362	350,362	350,362	350,362
Week FE		✓	✓	✓	✓
Role FE			✓	✓	✓
Sector FE				✓	✓
Seniority FE					✓

Notes – This table reports results of regressions of relative autocompletion accept rates on work experience. The columns progressively add week, role, sector, and seniority fixed effects. One standard deviation in work experience corresponds to approximately 7 years of experience. Standard errors clustered by user are reported in parentheses.

A.4. Additional Exhibits on Merge Output

I include additional exhibits related to the analysis of merge output.

- Table A18 reports characteristics of firms in the outcomes sample and firms in the rest of the sample.
- Table A19 reports pre-period balance statistics for the merge metadata sample.
- Figure A20 plots raw outcomes across the groups.
- Table A21 reports period-by-period means for the eligible and baseline groups.
- Table A22 reports results using the wild cluster bootstrap-t.
- Table A23 reports results using the log of output metrics as dependent variables.
- Table A24 reports heterogeneity results using the log of merges as the dependent variables.
- Table A25 reports heterogeneity analysis across number of users.
- Table A26 reports additional estimates related to lines edited per merge and number of files touched per merge.
- Table A27 reports subsample analyses by partitions of the eligible group.

Characteristics table Table A18 compares characteristics of firms in the outcomes sample and firms in the rest of the sample.

Table A18: Characteristics across samples.

	Eligible Group (pre beta)	Eligible Group (endline)	Baseline Group (endline)	Rest of Sample (endline)
N users	115.8	393.3	119.1	113.2
Years of experience	11.1	11.1	12.0	12.0
Engineer share	0.73	0.67	0.77	0.73
N firms	24	24	8	968

Notes – This table reports means of firm-level means across the eligible group, baseline group, and rest of sample. Endline statistics are computed with respect to the final user join date present on the sample, which is August 31, 2025. The pre-beta eligible group statistics reflect users on the platform who joined before the beta release.

Balance table Table A19 is a balance table for the difference-in-differences specification.

Table A19: Balance table.

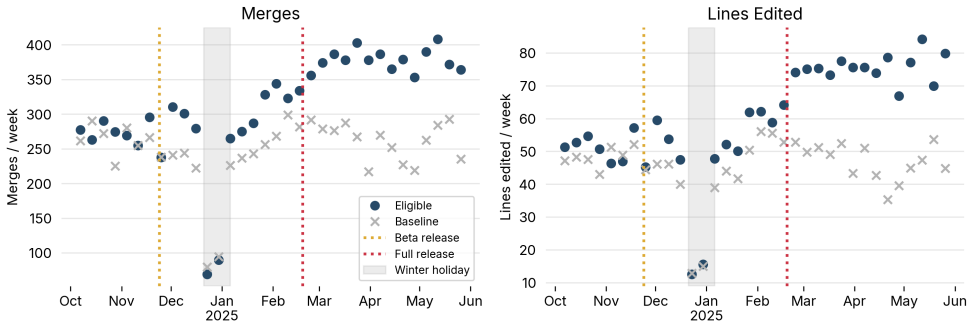
	Eligible Group	Baseline Group	Normalized Difference
Merges	271.7	264.3	0.04
Lines Edited (K)	50.5	47.7	0.07
Bugfix Rate (%)	10.4	10.0	0.06
Revert Rate (%)	1.07	0.87	0.20
Lines Edited / Merge	197.0	195.9	0.02
Files Touched / Merge	6.0	6.1	-0.11
Organizations	24	8	

Notes – This table reports pre-initial release means of the main outcome measures for the eligible and baseline groups. The eligible group contains organizations that adopted the Cursor platform before the sample period began in October 2024. The baseline group contains organizations that did not use the Cursor platform during the sample period. Note that the average of lines edited / merge (a mean of ratios) is greater than average lines edited / average merges (a ratio of means) since weekly merges and lines edited per merge are negatively correlated.

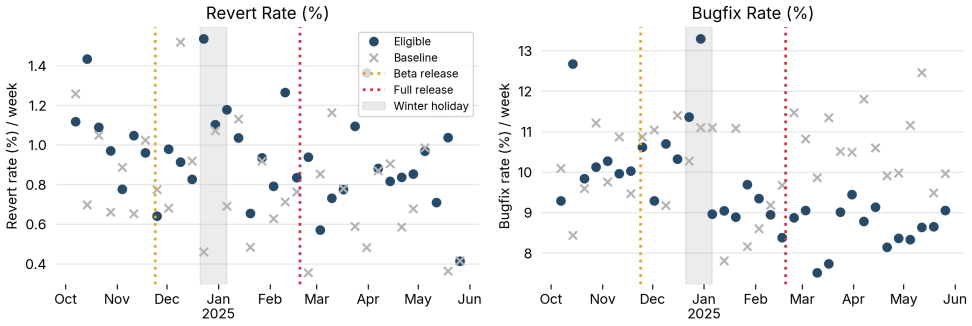
Raw outcomes Figure A20 plots mean outcomes across the groups. There is seasonality in merges and lines edited during holidays, which is adjusted for in the results that use the difference-in-differences specification.

Figure A20: Outcomes surrounding the agent feature release.

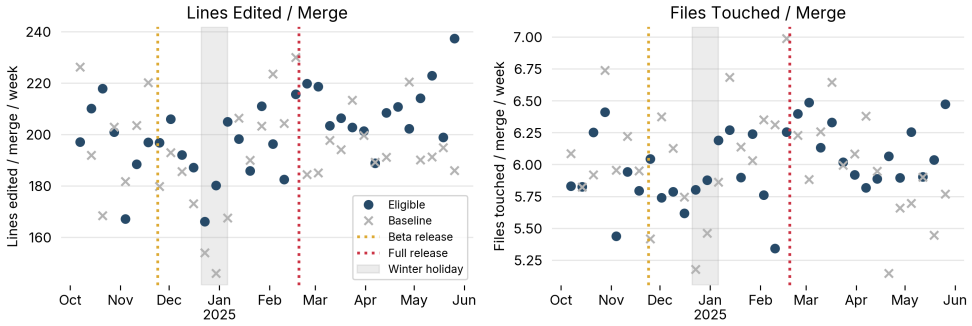
(a) Output (larger outcomes indicate higher output).



(b) Quality (smaller outcomes indicate lower rates of short-run issues).



(c) Scope (larger outcomes proxy for larger-scoped merges).



Notes – This figure plots mean outcomes related to throughput (merges and lines edited), quality (bugfix rate and revert rate), and scope (lines edited per merge and files touched per merge) for organizations in the eligible and baseline groups. Vertical lines denote the release date and default date of the agent feature.

Outcome table Table A21 reports means of the output and quality measures across release periods and organization groups.

Table A21: Mean outcomes across groups and periods.

Period	Group	Merges	Lines Edited (K)	Revert Rate (%)	Bugfix Rate (%)
Pre Beta Release	Eligible	271.7	50.5	1.07	10.4
	Baseline	264.3	47.7	0.87	10.0
Post Beta Release (Complete Window)	Eligible	322.7	62.4	0.92	9.3
	Baseline	245.2	44.8	0.79	10.3
Post Beta Release & Pre Full Release	Eligible	262.1	48.1	0.99	10.0
	Baseline	224.5	41.8	0.85	9.9
Post Beta Release & Post Full Release	Eligible	375.2	74.8	0.86	8.6
	Baseline	263.1	47.4	0.73	10.6

Notes – This table reports mean outcomes across periods for the eligible and baseline groups.

Bootstrap Table A22 reports estimates from the main difference-in-differences specification. Inference uses the wild cluster bootstrap-t (Cameron et al., 2008), clustered by organization and implemented via the `fwildclusterboot` package (Fischer and Roodman, 2021). The analysis is computed using 9,999 replications with Rademacher weights, absorbing week fixed effects. Two-sided p-values and 95% confidence intervals are reported. The original t-statistic from clustering by organization is also reported. There are 24 eligible organizations and 8 baseline organizations.

Table A22: Results from bootstrap inference.

(a) Post beta versus pre beta.

Outcome	Estimate	p-value	CI Low	CI High	t-statistic
Merges	70.09	0.002	31.74	108.63	3.77
Lines Edited (K)	14.80	0.006	4.68	24.70	3.01
Revert Rate (%)	-0.074	0.595	-0.361	0.212	-0.54
Bugfix Rate (%)	-1.41	0.030	-2.66	-0.16	-2.42

(b) Post full release versus pre beta.

Outcome	Estimate	p-value	CI Low	CI High	t-statistic
Merges	104.69	0.002	52.99	156.36	4.10
Lines Edited (K)	24.61	0.003	9.36	39.85	3.29
Revert Rate (%)	-0.081	0.624	-0.429	0.274	-0.49
Bugfix Rate (%)	-2.40	0.004	-3.85	-0.92	-3.43

Notes – These tables report the results of the main difference-in-differences regressions, along with wild cluster bootstrap p-values and confidence intervals.

Regressions with log of output metrics Table A23 reports results from the main difference-in-differences specification with log merges and log lines edited as dependent variables.

Table A23: Output results using log outcomes as dependent variables.

	Post Beta Release		Post Full Release	
	log(Merges)	log(Lines Edited)	log(Merges)	log(Lines Edited)
	(1)	(2)	(3)	(4)
Eligible × Post	0.24**	0.33***	0.36***	0.46***
	(0.09)	(0.12)	(0.10)	(0.12)
R^2	0.87	0.80	0.92	0.84
Within R^2	0.01	0.01	0.05	0.05
Observations	1,088	1,088	672	672
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓

Notes – This table reports regression results with log merges and log lines edited as dependent variables. Standard errors are clustered by firm.

Heterogeneity with log of merges Table A24 reports the results corresponding to heterogeneous increases in output, following Table 12. The dependent variable is replaced with the natural logarithm of merges.

Table A24: Heterogeneity results using log merges as the dependent variable.

(a) Post beta versus pre beta.

	Software Engineer Share		Average Work Experience	
	Low	High	Low	High
Eligible × Post Beta Release	0.29***	0.19	0.22*	0.27**
	(0.10)	(0.12)	(0.12)	(0.11)
R^2	0.89	0.86	0.87	0.88
Within R^2	0.03	0.01	0.01	0.02
Observations	680	680	680	680
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓

(b) Post full release versus pre beta.

	Software Engineer Share		Average Work Experience	
	Low	High	Low	High
Eligible × Post Full Release	0.41***	0.30**	0.33**	0.39***
	(0.12)	(0.13)	(0.14)	(0.12)
R^2	0.91	0.93	0.91	0.93
Within R^2	0.10	0.05	0.06	0.09
Observations	420	420	420	420
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓

Notes – These tables report the results of the heterogeneous effects regressions with log merges as the outcome variable. Standard errors clustered by firm are reported in parentheses.

Heterogeneity across size of user base Table A25 reports heterogeneity analyses that split the eligible group across the median of number of users in the pre-beta-release period.

Table A25: Heterogeneity across size of user base.

(a) Heterogeneity across size of user base (merges).

	Post Beta Release Number of Users		Post Full Release Number of Users	
	Low	High	Low	High
Eligible × Post	63.04***	77.14**	90.33***	119.06**
	(21.63)	(27.59)	(27.09)	(42.13)
R^2	0.85	0.88	0.87	0.91
Within R^2	0.03	0.02	0.09	0.08
Observations	680	680	420	420
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓
Pre-Beta Mean	208.4	335	208.4	335
Percent Change	30.2%	23.0%	43.3%	35.5%

(b) Heterogeneity across size of user base (log merges).

	Post Beta Release Number of Users		Post Full Release Number of Users	
	Low	High	Low	High
Eligible × Post	0.25**	0.24*	0.40***	0.32**
	(0.10)	(0.13)	(0.10)	(0.15)
R^2	0.83	0.90	0.90	0.93
Within R^2	0.02	0.01	0.10	0.05
Observations	680	680	420	420
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓

Notes – These tables report heterogeneity analyses split by medians of pre-beta-release number of users. Standard errors are clustered by firm.

Merge scope Table A26 reports the results of regressions following

$$Y_{i,t} = \alpha_i + \alpha_t + \beta_1 \times \text{Eligible}_i \times \text{Post Beta}_t + \beta_2 \times \text{Eligible}_i \times \text{Post Full}_t + \varepsilon_{i,t} \quad (6)$$

These regressions present subperiod analyses that divide the post-release period into a pre-full release and post-full release period.

The regressions include outcome variables related to scope: Lines edited per merge and files touched per merge. Analogously to lines edited, number of files touched per merge is winsorized at the 1% level at the right tail, and then averaged to the weekly level. The merge scope subsample analysis considers changes in small and large merges, defined as those below or above median lines edited or files touched for the eligible group in the pre-beta period.

Table A26a does not find evidence of changes in lines edited per merge or files touched per merge. Table A26b indicates that there were increases in both small and large merges. In the eligible group, pre-release means of merges per week were 137.0 for small merges and 134.5 for large merges (by lines edited), and 146.1 for small merges and 125.6 for large merges (by files touched).

In the initial period of agent use, there is directional evidence that eligible organizations produced more merges of both types. Changes are computed by dividing each $\hat{\beta}_1$ by pre-release means. When scope is defined by lines edited, eligible organizations produced small merges 12% more often and large merges 10% more often. When scope is defined by files touched, eligible organizations produced small merges 10% more often and large merges 12% more often.

In the default period, eligible organizations produced more merges of both types. Changes are computed by dividing each $\hat{\beta}_1 + \hat{\beta}_2$ by pre-release means. When scope is defined by lines edited, eligible organizations produced small merges 33% more often and large merges 44% more often. When scope is defined by files touched, eligible organizations produced small merges 39% more often and large merges 38% more often.

Table A26: Estimates from subperiod and subsample analyses.

(a) Estimates from subperiod specification across outcomes.

	Merges (1)	Lines Edited (K) (2)	Revert Rate (%) (3)	Bugfix Rate (%) (4)	Lines / Merge (5)	Files / Merge (6)
Eligible × Post Beta Release	30.17*	3.47	-0.07	-0.27	3.29	0.08
	(16.89)	(3.33)	(0.15)	(0.69)	(20.85)	(0.43)
Eligible × Post Full Release	74.52***	21.14***	-0.02	-2.12***	10.48	0.22
	(23.37)	(6.63)	(0.15)	(0.75)	(12.48)	(0.26)
R^2	0.87	0.83	0.27	0.53	0.50	0.50
Within R^2	0.04	0.04	0.00	0.01	0.00	0.00
Observations	1,088	1,088	1,088	1,088	1,088	1,088
Organization FE	✓	✓	✓	✓	✓	✓
Time FE	✓	✓	✓	✓	✓	✓

(b) Estimates from subperiod specification across merge scope subsamples.

	Merges			
	Small (Lines) (1)	Large (Lines) (2)	Small (Files) (3)	Large (Files) (4)
Eligible × Post Beta Release	16.84*	13.38	14.95*	15.22
	(8.46)	(9.99)	(8.30)	(10.64)
Eligible × Post Full Release	27.82***	46.31***	41.70**	32.82***
	(9.06)	(15.79)	(15.37)	(10.22)
R^2	0.88	0.86	0.86	0.87
Within R^2	0.03	0.04	0.03	0.04
Observations	1,088	1,088	1,088	1,088
Organization FE	✓	✓	✓	✓
Time FE	✓	✓	✓	✓

Notes – These tables report difference-in-differences estimates of outcomes among organizations eligible for the agent feature. The estimates follow the specification in Equation (6). The top panel reports estimates related to output, quality, and scope. The bottom panel compares estimates filtering to merges above and below the median of lines edited or files touched. Standard errors clustered by organization are reported in parentheses.

Effects by organization subsamples Table A27 reports variations of the main difference-in-differences results that split the eligible group across pre-release outcomes. These outcomes are weekly merges, weekly lines edited, revert rates, and bugfix rates. Eligible organizations are ranked by pre-release means of each outcome, and then split along the median. The results indicate that eligible organizations with higher pre-release revert rates had no higher increases in merges after the agent was released than eligible organizations with lower pre-release revert rates.

Table A27: Subsample analysis by pre-release merge characteristics.

(a) Post beta versus pre beta.

Outcome for Subsample Split:	Merges		Lines Edited		Revert Rate		Bugfix Rate	
Eligible Group Subsample:	Low	High	Low	High	Low	High	Low	High
Dependent Variable:	Merges							
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Eligible × Post Beta Release	45.45***	94.74***	62.73***	77.45**	73.94***	66.25**	72.00**	68.19***
	(13.94)	(30.52)	(18.34)	(29.86)	(25.41)	(24.28)	(27.02)	(22.52)
R^2	0.88	0.84	0.88	0.85	0.88	0.86	0.85	0.90
Within R^2	0.03	0.03	0.04	0.02	0.02	0.03	0.02	0.03
Observations	680	680	680	680	680	680	680	680
Organization FE	✓	✓	✓	✓	✓	✓	✓	✓
Time FE	✓	✓	✓	✓	✓	✓	✓	✓
Pre-beta mean merges:	101.29	442.17	122.60	420.86	261.50	281.96	320.29	223.17
Percent change:	44.9%	21.4%	51.2%	18.4%	28.3%	23.5%	22.5%	30.6%

(b) Post full release versus pre beta.

Outcome for Subsample Split:	Merges		Lines Edited		Revert Rate		Bugfix Rate	
Eligible Group Subsample:	Low	High	Low	High	Low	High	Low	High
Dependent Variable:	Merges							
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Eligible × Post Full Release	51.05***	158.33***	74.63***	134.76***	113.01**	96.38***	113.03***	96.36**
	(17.55)	(41.32)	(24.93)	(41.94)	(39.74)	(30.90)	(35.52)	(35.67)
R^2	0.90	0.88	0.90	0.88	0.91	0.89	0.88	0.92
Within R^2	0.05	0.12	0.07	0.09	0.08	0.08	0.07	0.09
Observations	420	420	420	420	420	420	420	420
Organization FE	✓	✓	✓	✓	✓	✓	✓	✓
Time FE	✓	✓	✓	✓	✓	✓	✓	✓
Pre-beta mean merges:	101.29	442.17	122.60	420.86	261.50	281.96	320.29	223.17
Percent change:	50.4%	35.8%	60.9%	32.0%	43.2%	34.2%	35.3%	43.2%

Notes – This table reports difference-in-differences estimates of merges among subsamples of organizations eligible for the agent feature. In each regression, the eligible group is split across medians of pre-release outcome variables. The estimates follow the specification in Equation (4). Standard errors clustered by organization are reported in parentheses.