Venture Capital and Engineering Quality: Evidence from Software Startups*

João Dimas† Lampros Pnevmatikos‡

September 21, 2026

Abstract

We investigate a novel channel of venture capital value creation: the practices through which startups build and maintain their technology. Analyzing more than 60,000 full codebase snapshots from over 550 firms, we compare changes in engineering quality between firms receiving their first institutional VC investment and similar firms without a first institutional round through the event window. We also construct a dollar-denominated measure of technical debt, the estimated maintenance liability embedded in software. Relative to codebase size, serious deviations from coding best practices decline by 15% and estimated maintenance liabilities by 12% of their respective treated pre-investment means. These improvements accompany substantial codebase growth. The decline in the frequency of serious coding problems persists after matching on pre-financing growth, and quality gains extend to firms already generating revenue. In contrast, a placebo test around later financing rounds shows no statistically significant quality improvement. After first entry, individual code contributions reveal more documented review and issue tracking, less off-hours work, and a smaller share of contributions by existing team members linked to later bug fixes. The findings suggest that VC professionalization extends to the practices through which existing teams build and maintain their products.

JEL Classification: G24, G34, D82, L86, L26
Keywords: Venture capital, technical debt, software engineering quality, code quality, difference-in-differences

*We are grateful to Margarita Tsoutsoura and Todd Gormley for their invaluable comments and guidance throughout this project. We also thank Julie Fu, Armando Gomes, Gerard Hoberg, Arthur Korteweg, Linda Schilling, Anjan Thakor, and participants in the 12th Annual USC Marshall Ph.D. Conference in Finance for their insightful feedback and suggestions. This paper was previously circulated with the title : Venture Capital, Engineering Quality, and Technical Debt. We acknowledge support from the Google Research Program, and Pnevmatikos thanks the Onassis Foundation for support. The authors have no other relationships or financial interests to disclose.

†Olin Business School at Washington University in St. Louis.

‡Olin Business School at Washington University in St. Louis. Send correspondence to Lampros Pnevmatikos; e-mail: l.pnevmatikos@wustl.edu.

1 Introduction

An important question in corporate finance concerns how venture capitalists contribute to the success of the startups they finance (Gompers et al., 2020; Amornsiripanitch, Gompers, and Xuan, 2019; Hsu, 2004). While venture capitalists select promising firms, their involvement after investment can also shape firm performance. For example, Bernstein, Giroud, and Townsend (2016) show that reductions in the cost of monitoring portfolio companies improve performance, especially innovation. Prior work documents that VCs reshape governance structures, leadership, and organizational design, but a fundamental operational margin remains unexplored: do venture capitalists also change the *technical practices* through which startups build their products?

We examine whether institutional entry also professionalizes *technical practice*—the engineering processes that govern product quality and scalability—a margin that, to our knowledge, prior work has not studied directly. Existing evidence on VC-driven professionalization focuses on organizational and governance channels: Hellmann and Puri (2002) document that VC-backed firms are more likely to hire professional CEOs, implement stock option plans, and formalize HR policies, and that these changes occur earlier than in comparable non-VC-backed firms. These are important operational changes in leadership and organizational structure, but they do not speak to whether VCs also reshape the day-to-day development practices that govern product quality and scalability.

For technology startups—a central part of the venture capital ecosystem—long-run value depends not only on strategy and governance, but also on the quality of the underlying software. Architectural choices and development practices drive the accumulation of technical debt (TD)—the future cost of remediating code issues—and can shape a startup's ability to scale reliably, iterate quickly, and manage operational risk as the organization grows (Sculley et al., 2015, 2014).

Despite its centrality in software engineering, technical debt (TD) has not yet been studied directly in the finance literature; to our knowledge, this paper is the first to do so. TD is not recorded in financial statements, yet it behaves like an economically meaningful liability. Deviations from engineering best practices can increase the future resources required to maintain and modify a product, and, if left unaddressed, can compound into higher defect risk and slower feature development, similar to postponing CapEx (Banker, Liang, and Ramasubbu, 2021; Krasner, 2022; McKinsey and Co, 2020, 2023). However, technical debt is notoriously hard to measure and compare across firms (Banker, Liang, and Ramasubbu, 2021; Krasner, 2022). In that sense, TD is closer to an off-balance-sheet operating liability than something that should only concern engineers. Our setting allows us to bring the industry-standard concept of TD into empirical finance by constructing a consistent, dollar-denominated measure from code-level data.

To accomplish this, we assemble longitudinal codebase data for 569 open-core technology firms. These firms commercialize software whose core source code is publicly available, allowing us to observe development histories for an economically relevant part of their products. We combine

more than 60,000 monthly snapshots of the entire codebase with financing records from PitchBook and employment histories from LinkedIn. Our focal event is the first completed Seed round of the firm, or the first completed Series A round when no Seed is recorded.

VC contracting and monitoring provide a rationale for changes in technical practice after financing. Staged capital provision gives investors opportunities to review progress and condition future funding on performance (Gompers, 1995), while control and board rights facilitate intervention (Kaplan and Strömberg, 2003). Investor advice can also affect entrepreneurial decisions when both parties contribute to the venture (Casamatta, 2003). In software firms, these activities may encourage more systematic review, testing, and coordination of development work. Financing can also fund engineers and tools directly. The net effect on quality is uncertain, however, because faster development and larger teams can increase complexity and coordination costs.

To answer this question, we estimate a stacked difference-in-differences model comparing changes in newly funded firms with contemporaneous changes in firms that remain without a first round throughout the event window. We measure coding problems relative to codebase size so that firms with different amounts of software can be compared. The frequency of serious deviations from coding best practices falls by 15% of the treated pre-investment mean. These are problems that can make software less stable, less secure, or more costly to maintain. We also find fewer problems that make software harder to modify or extend. Improvements in minor code issues and in the separate measures of software reliability and security are not statistically significant.

Our technical-debt measure translates these coding problems into estimated future maintenance costs. Relative to codebase size, this liability falls by 12% of the treated pre-investment mean. The decline therefore represents a lower estimated maintenance burden for a given amount of software.

Event-study plots show that serious coding problems become less frequent around the first financing round, with little divergence between treated and control firms before funding. This timing pattern is consistent with a change in development practices accompanying investor entry. The decline occurs as codebases expand substantially, while the estimated change in the total number of serious coding problems is not statistically significant. A growing codebase can have fewer problems per unit of code even if existing problems are not removed, provided that problem counts grow more slowly than code volume. These comparisons alone therefore cannot distinguish the correction of existing problems from the addition of code with fewer problems.

We examine whether the finding persists under alternative ways of accounting for codebase size. First, the decline in serious coding problems remains among firms with slow pre-financing code growth. Second, we model the number of problems directly, allowing problem counts to rise faster or more slowly than codebase size. Funded firms have fewer serious coding problems in this comparison as well. Because financing can itself affect codebase size, this second result describes a comparison conditional on product scale; it complements the decline in problems per unit of code.

We also acknowledge that VC funding is inherently selective and non-random: investors finance

firms they expect to grow. A natural concern is that the quality improvements reflect growth trajectories that were already different before financing. Although engineering-quality measures are well balanced before funding, treated firms are smaller and grow faster than controls. To address this concern, we match treated and control firms on pre-financing headcount growth within each financing cohort. The declines in serious coding problems and problems that make software harder to maintain persist across a range of matching choices. The estimated reduction in maintenance liabilities also remains similar in magnitude, although it is no longer statistically significant in the smaller matched sample. These comparisons show that the main coding-quality findings persist when newly funded firms are compared with startups on similar prior growth trajectories.

To examine whether quality improvements recur with additional financing, we implement a placebo test centered on each firm's first Series B-or-later round. Around these later rounds, total headcount and software-engineering headcount rise substantially, as they do around first VC entry. Yet we find no statistically significant improvement in coding quality or estimated maintenance liabilities. This combination of substantial hiring and no detectable quality gains is consistent with a change in technical practice associated with first institutional entry.

We also test whether the quality improvements reflect a transition to commercialization around the first financing round. Most treated firms were already generating revenue before financing, and quality gains extend to these firms. We find no evidence of larger improvements among firms that had not yet begun selling their products, or of systematic differences between initial financing stages. Development activity expands, and employment grows in both engineering and commercial functions, with a larger shift toward engineering. However, developers' descriptions of code changes do not show a statistically significant increase in references to commercialization, such as pricing, subscriptions, or billing. The quality improvement is therefore not confined to firms making the initial transition to revenue generation.

We then examine the development process directly using 4.8 million individual code contributions. Following first VC entry, a larger share of contributions carries a documented review or sign-off, and larger shares are linked to tracked tasks or bugs and to proposals for code changes submitted for review. At the same time, a smaller share of contributions takes place outside regular working hours or on weekends. Tracing later bug fixes back through code histories also reveals that a lower fraction of contributions by engineers already present before financing is linked to subsequent fixes. Together, these findings document more formal development procedures and suggest improvements in the work of existing teams, providing evidence on how engineering practices change around institutional entry.

Our results contribute to several strands of the literature. First, we contribute to the venture capital literature on post-investment value creation. While prior work has emphasized governance and leadership professionalization, we provide evidence on the technical practices through which startups build their core products. Our findings suggest that coding discipline improves around VC entry, with reductions in serious problems and in problems that make software difficult to maintain.

Second, the results complement evidence on "operational engineering" in buyout markets (Bernstein and Sheen, 2016). We document changes in production practices at an earlier stage of firm development, when the first institutional investor enters. Later rounds accompany substantial hiring without a statistically significant quality gain. Read alongside the contribution histories, this difference supports a governance interpretation in which first institutional entry strengthens technical discipline.

Third, our measurement of software maintenance liabilities informs research on intangible assets and software-intensive firms (Eisfeldt and Papanikolaou, 2013; Peters and Taylor, 2017). We construct, to our knowledge, the first comparable, dollar-denominated measure of technical debt in the finance literature. Combining this measure with development histories allows researchers to study how firms build and maintain software assets, with applications to technology-firm valuation and M&A due diligence.

The remainder of the paper proceeds as follows. Section 2 describes the data, sample construction, and measures of engineering quality and technical debt. Section 3 presents the empirical design, results, and mechanism evidence. Section 4 examines matching, later financing rounds, commercialization, and specification sensitivity. Section 5 concludes.

2 Data & Background

We combine three data sources to follow software development, employment, and financing at open-core technology firms: **(i)** longitudinal codebase snapshots from the GitHub and GitLab application programming interfaces (APIs); **(ii)** employee–firm matched data from LinkedIn; and **(iii)** funding and deal records from PitchBook.

Firms typically restrict access to proprietary source code, making their software assets difficult to study. We therefore focus on open-core firms: they make the core of their product publicly available, allowing us to observe its development history and technical choices.

2.1 The Open-Core Business Model

The open-core model combines the collaborative advantages of open-source software with the revenue potential of proprietary licensing. Under this model, a firm releases its core product under an open-source license, making the source code freely accessible to developers and researchers. The firm retains exclusive rights to enterprise-grade features, proprietary modules, or support services, which it monetizes through subscription or licensing agreements.

A prominent example is Posit (formerly RStudio, Inc.), the company behind the RStudio integrated development environment for the R programming language. While the core software is free and open source, the company generates revenue by licensing enhanced versions for commercial applications. This approach is widespread in the technology industry and draws on community con-

tributions to accelerate market entry (Fershtman and Gandal, 2011; Nagle, 2019). Public source code lets us observe a firm's development history and technical choices, which are typically unavailable to researchers.

The open-core setting also has implications for external validity. Relative to closed-source startups, open-core firms expose substantial portions of their codebase (and often issue trackers) to public scrutiny, which can create baseline quality discipline through reputational concerns, external bug reporting, and community review. If this community discipline substitutes for some of the monitoring and process improvements that VCs bring, the marginal effect of VC entry on engineering quality could be attenuated in our setting and potentially larger in proprietary environments. Conversely, transparency may complement VC involvement by making engineering performance more observable and by facilitating rapid community-driven fixes, so the direction of any external-validity bias is not a priori clear. We therefore interpret our estimates as evidence that VC entry is associated with changes in engineering outcomes in an observable, open-core environment.

2.2 Codebase Data

We define our sample universe using OSSrank, a leading analytics platform for the open-source ecosystem, to identify 649 unique companies that commercialize open-source repositories. These firms primarily operate in the technology sector.

Using the GitHub and GitLab APIs, we match each firm to the repositories that hold the code underlying its core product. A repository is the central storage location for a software project's source code. GitHub and GitLab are industry-standard platforms for software development and version control, widely adopted across both startups and large enterprises. These platforms record saved modifications to the code as timestamped "commits" containing rich metadata: the author, the precise nature of the change, and an accompanying message.

We construct a monthly panel of codebases by retrieving the final snapshot of each repository for every month in our sample period (2010–2024). We analyze these snapshots with SonarQube using a fixed scanner version and configuration, applying a consistent measurement procedure across firms and over time. While most repositories are hosted on GitHub, some firms use GitLab. Since both platforms offer nearly identical version control features, we pool these data sources into a combined dataset.

Because a firm can maintain several repositories, we measure engineering quality at the firm–month level by aggregating across its available repositories. We sum lines of code, remediation effort, and issue counts, and recompute ratio measures from these totals. This construction captures development across the firm's covered projects while retaining the firm as the unit of analysis. Table 17, Panel B, reports a specification that excludes firms with more than five scanned repositories.

The resulting panel links 703 repositories to 621 firms, 569 of which contribute at least one

covered codebase month. We use 61,624 repository–month SonarQube snapshots and fill gaps between observed months by linearly interpolating metrics within firms. The monthly panel contains 61,768 firm–months with a positive line count. For estimation, we retain months with a recorded scan and a positive line count. The stacked estimation sample contains 484 unique firms and 99 financing cohorts: 274 firms appear as treated and 412 as controls. These counts overlap because a firm can serve as a control before its own financing event.

2.3 Funding & Firm Data

We date VC entry using PitchBook financing records. For each firm, we extract its investment history, including deal dates, transaction types, investor composition, and funding stages. We first remove rounds recorded as rumored, failed or cancelled, announced or in progress, upcoming, or postponed. We then identify the first completed Seed round, or the first completed Series A round when no completed Seed is recorded. Of the 274 treated firms, 202 enter at Seed and 72 at Series A. To focus on early-stage venture investment, we exclude firms that enter the data only through a public listing, debt issuance, management buyout, acquisition, or Series B or later round with no prior Seed or Series A financing. We separately examine later financing rounds in our Series B+ placebo tests.

Since the firms in our sample are predominantly private and early stage, standard financial disclosures and coverage in databases such as Compustat are limited. To measure firm size and growth, we use employee–firm matched LinkedIn records to construct monthly snapshots of total headcount and software-engineering headcount. These histories track organizational expansion alongside changes in the codebase.

2.4 Measuring Engineering Quality

We measure engineering quality using SonarQube, a widely used static-analysis tool that scans source code and flags deviations from common coding and security practices. SonarQube is typically run on the current version of a repository during development, whereas our empirical design requires the same measures for past versions. We therefore run SonarQube retrospectively on historical snapshots to construct a firm–month panel of engineering-quality outcomes.

For each firm and month in our sample period (2010–2024), we (i) identify the repository state at month-end, (ii) scan that codebase with SonarQube, and (iii) record the metrics in a panel dataset. Across firms and months, this produces more than 61,000 distinct codebase snapshots. We store outputs in our own research database because SonarQube Community Edition[1] is designed for monitoring a small number of live projects rather than retaining a large archive of historical scans.

A major challenge in analyzing diverse open-core repositories is handling multiple programming

[1] SonarQube Community Edition is the freely available, open-source distribution used for all of our scans.

languages consistently, especially when historical build environments are hard to reproduce. For interpreted languages such as Python, JavaScript, and Ruby, SonarQube analyzes source code directly. For compiled languages such as Java, C, and C++, historical revisions often do not build cleanly, which can break standard analysis workflows. To address this difficulty, we configure the analysis to operate directly on source files and automatically detect and scan all languages present in each repository.

Analyzing source files keeps measurement consistent over time. We scan C/C++ repositories without functional build scripts and Java applications from their `.java` files even when compiled `.class` files are missing. Applying language-specific rules through the same procedure supports comparisons across firms using different programming languages. Appendix B details the configurations for selected languages.

2.4.1 Taxonomy of Engineering Defects

The SonarQube scan checks code against thousands of language-specific rules that reflect widely used software-quality frameworks and industry practice (e.g., ISO/IEC 25010). Each time the code triggers a rule, SonarQube records an "issue"; these issues are the basic unit behind our software quality measures. Three common Java rule violations illustrate what the tool measures. (i) Commented-out code makes it harder to understand what the software actually does and whether the implementation is complete. (ii) Lingering `TODO` markers are placeholders for unfinished work that often persist and embed a hidden backlog inside the code. (iii) Overly complex functions (high cognitive complexity) are harder to read, test, and modify, raising the cost of future changes and increasing the risk of mistakes.[2]

We group detected issues into three domains. Reliability covers code patterns that can cause failures or unexpected behavior, such as attempts to access nonexistent objects or loops that never terminate. Security covers weaknesses such as exposed credentials or unsafe handling of database queries. We also track *Security Hotspots*, which flag sensitive code for manual review. Maintainability concerns the ease of modifying or extending the code, including problems such as duplication and complex control flow.

SonarQube assigns each issue a severity level based on its potential impact.[3] In the empirical analysis, we focus on *Critical Violations*—severe deviations from coding best practices with a potentially substantial impact on reliability, security, or future maintainability. SonarQube assigns these issues high remediation priority. We distinguish them from lower-severity violations (*Major, Minor*) to separate more consequential engineering problems from incremental quality improvements.

[2] For additional examples of common Java issues highlighted by SonarSource (e.g., duplicated string literals, unused elements, raw types, generic exceptions), see SonarSource, "Top issues in Java projects".

[3] SonarQube's severity taxonomy includes a *Blocker* category above Critical for issues requiring immediate attention. We retain the distinction between these categories and focus on Critical Violations in the main analysis.

2.5 Technical Debt

Technical Debt (TD) measures the stock of maintenance liabilities embedded in a codebase. In the software engineering literature, technical debt captures development choices that save time today but raise the cost of future changes (Sculley et al., 2014). Firms effectively pay interest through additional maintenance work and face costly paydowns when refactoring—reorganizing existing code without changing its intended functionality—becomes unavoidable. For example, duplicating code can accelerate a feature's introduction but requires developers to update several copies whenever that feature changes. Although this tradeoff is familiar in software engineering, it has received little direct measurement in finance; our approach expresses these software liabilities in dollars so they can be compared across firms and over time.

To measure TD, we use SonarQube's SQALE (Software Quality Assessment based on Lifecycle Expectations) methodology (Letouzey, 2012). SonarQube flags maintainability issues, such as excessive complexity or duplicated code, and assigns each issue a model-implied remediation time. Aggregating these times across all issues in a firm's covered repositories yields the firm–month SQALE Index, measured in hours.[4] This measure weights maintainability issues by the estimated effort required to resolve them.

The raw SQALE Index measures remediation effort under a fixed scan configuration, but its absolute level depends on the enabled rules and their assigned remediation times. We therefore apply a common scaling factor, κ, to calibrate the index to an external economic benchmark. This produces a dollar-denominated proxy for maintenance liabilities, with a consistent unit of measurement across firms and monthly snapshots. The calibration concerns the level of the liability; it does not assume that firms incur the modeled remediation costs as current cash expenditures.

The calibration proceeds in three steps. First, we construct a proxy for the replacement cost of the observed software assets. Let ℓ denote development time per non-comment line of code. We set $\ell = 0.5$ hours (30 minutes), the default parameter in SonarQube's development-cost calculation. For a codebase with $\text{LoC}_{i,t}$ lines of code, our proxy for replacement effort is:

$$\text{Re-development Effort}_{i,t} = \ell \times \text{LoC}_{i,t}. \tag{1}$$

Multiplying by an assumed common developer wage of $w = \$55$ per hour yields Software Assets in dollars:

$$\text{Software Assets}_{i,t} = w \times \text{Re-development Effort}_{i,t}. \tag{2}$$

Second, we choose the scaling factor κ to place these remediation estimates on an economic scale. Surveyed technology executives estimate technical debt at 20–40% of the value of their technology

[4] SonarQube stores remediation time in minutes; we convert it to hours.

assets before depreciation (McKinsey and Co, 2023). We use the lower endpoint as a calibration target, setting median debt equal to 20% of the median replacement-cost proxy.[5] Letting $\varphi = 0.20$, we choose κ such that:

$$\text{Median}\big(w \times \kappa \times \text{SQALE}_{i,t}\big) = \varphi \times \text{Median}\big(w \times \ell \times \text{LoC}_{i,t}\big)\,, \tag{3}$$

where we compute medians over nonmissing firm–month observations in the source panel. Simplifying yields:

$$\kappa = \varphi \times \frac{\text{Median}\big(\ell \times \text{LoC}_{i,t}\big)}{\text{Median}\big(\text{SQALE}_{i,t}\big)}. \tag{4}$$

In the main sample, this produces $\kappa = 68.60$, which is constant across firms and time.

Third, we apply this calibration to construct Technical Debt for each firm–month:

$$\text{Technical Debt}_{i,t} = w \times \kappa \times \text{SQALE}_{i,t}. \tag{5}$$

Because κ is constant across firms and time, it rescales the SQALE Index proportionally, leaving statistical significance and changes relative to the sample mean unchanged. The calibration makes the estimated maintenance burden comparable in dollar terms, with its level determined by the common wage and calibration target.

We also report the Technical Debt Ratio, which expresses calibrated debt as a percentage of the replacement-cost proxy:

$$\text{Technical Debt Ratio}_{i,t} = 100 \times \frac{\text{Technical Debt}_{i,t}}{\text{Software Assets}_{i,t}} = 100 \times \frac{\kappa \times \text{SQALE}_{i,t}}{\ell \times \text{LoC}_{i,t}}. \tag{6}$$

The wage cancels, making the ratio independent of the assumed hourly wage, while its level reflects the calibration target.[6] Changes in the ratio, measured in percentage points, indicate how the maintenance burden evolves relative to the size of the software asset.

2.6 Key Variables

Our empirical analysis distinguishes engineering outcomes by both their economic consequence and their severity. We express issue counts per 1,000 non-comment lines of code (kLoC), allowing comparisons across firms with codebases of different sizes. *Critical Violations* capture deviations from coding best practices that SonarQube assigns high remediation priority because of their potential

[5] The survey covers broader technology assets than the source code we observe. We use it to calibrate the level of the measure.

[6] The calibration equates median Technical Debt with 20% of median Software Assets. The median firm–month ratio is 30.0%, since the median of a ratio need not equal the ratio of medians.

impact on the application. These can include risky code structures that make future modifications more error-prone. More broadly, *Violations (All)* count rule violations across every severity level, providing a broader measure of compliance with coding standards.

Maintainability Issues identify code patterns that make future modification or extension more costly, such as excessive complexity, duplication, and poor structural organization. High-severity issues have a larger assessed impact on maintainability and receive greater remediation priority; low-severity issues have a smaller assessed impact. By comparison, *Bugs* capture code patterns that are likely to produce incorrect or unexpected behavior, such as an invalid operation or a failure to handle an input correctly. The newer *Reliability Issues* measures assess related risks to software behavior, with high- and low-severity categories distinguishing their potential impact. These classifications overlap, so they provide complementary views of engineering quality rather than counts of mutually exclusive defects.

Finally, *Security Hotspots* identify security-sensitive code segments that require human review, such as code handling authentication or sensitive data. A hotspot marks a location that warrants inspection; it becomes evidence of a vulnerability only after the security context has been assessed. This distinction separates our measure of security-review needs from confirmed security failures.

In addition to these engineering measures, we use the economic measures of Technical Debt described in Section 2.5: debt in dollars per line of code and the Technical Debt Ratio, which scales debt by software assets. We measure codebase size using non-comment lines of code and organizational scale using total and software-engineering headcount, taking the natural logarithm of one plus each variable in the regressions. The VC indicator, $VC_{i,c,t}$, equals one for a firm treated in cohort c from its qualifying financing month onward. To examine heterogeneity, *High Violations* identifies firms whose critical violation density in the month before the cohort's event exceeds the median across baseline firm–cohort observations of firms treated somewhere in the sample; *Large Round* identifies financing rounds above the treated-firm median deal size. Table 19 in Appendix A provides the full variable definitions.

2.7 Summary Statistics

Table 1 reports summary statistics for the firm–month panel before stacking. The median covered firm–month has 5.16 critical violations per 1,000 lines of code and a Technical Debt Ratio of 29.98%. The distributions are right-skewed: the corresponding means are 10.45 and 43.47%.

3 Estimation & Empirical Results

To estimate how first institutional VC entry changes engineering quality, we exploit differences in financing dates across firms using a stacked difference-in-differences (DiD) design. With staggered

Table 1: Summary Statistics

	Distribution					
Variable	**N**	**Mean**	**SD**	**P25**	**Median**	**P75**
Panel A: Level Variables						
Technical Debt (in $)	61,768	5,263,622.30	11,987,074.44	164,190.37	938,639.44	4,201,349.26
Comment Lines	61,768	117,976.35	329,153.82	4,707.75	21,377.50	98,217.00
Total Violations	61,768	11,732.45	26,172.37	399.00	2,131.00	10,016.25
Critical Violations	61,768	4,133.78	9,799.54	83.00	619.00	3,210.00
Bugs	61,768	593.95	1,920.59	5.00	49.00	279.00
Security Hotspots	61,768	275.21	632.14	9.00	56.00	223.00
Maintainability Issues (Total)	61,768	11,097.16	24,925.69	375.00	2,038.00	9,395.25
Maintainability Issues (High)	61,768	4,106.40	9,747.16	79.00	608.00	3,196.00
Maintainability Issues (Low)	61,768	3,445.93	8,202.71	63.00	472.00	2,522.00
Reliability Issues (Total)	61,768	1,135.75	4,286.75	13.00	114.00	579.00
Reliability Issues (High)	61,768	28.33	75.54	0.00	3.00	20.67
Reliability Issues (Low)	61,768	463.55	3,109.39	0.00	11.00	134.00
Panel B: Per-LoC Variables						
Technical Debt per LoC ($)	61,768	11.92	15.99	3.55	8.29	14.26
Technical Debt Ratio (%)	59,903	43.47	58.69	12.92	29.98	51.85
Violations per 1,000 LoC	61,768	29.44	40.29	8.14	16.84	34.90
Critical Violations per 1,000 LoC	61,768	10.45	18.74	1.44	5.16	9.75
Bugs per 1,000 LoC	61,768	1.77	5.47	0.04	0.45	1.39
Security Hotspots per 1,000 LoC	61,768	1.37	10.60	0.10	0.43	1.05
Maintainability Issues (All) per 1,000 LoC	61,768	27.44	37.71	7.65	16.10	32.55
Maintainability Issues (High) per 1,000 LoC	61,768	10.35	18.68	1.37	5.09	9.67
Maintainability Issues (Low) per 1,000 LoC	61,768	7.80	15.76	1.24	3.86	7.92
Reliability Issues (Total) per 1,000 LoC	61,768	3.28	9.91	0.15	0.83	2.53
Reliability Issues (High) per 1,000 LoC	61,768	0.10	0.40	0.00	0.02	0.09
Reliability Issues (Low) per 1,000 LoC	61,768	1.36	5.73	0.00	0.09	0.51
Panel C: Scale Variables (Log)						
Log(Lines of Code)	61,768	11.78	2.07	10.66	12.05	13.22
Log(Headcount)	32,313	3.71	2.47	1.95	3.18	4.88
Panel D: Treatment Indicators (Post-Funding)						
Post Seed Round (=1)	111,780	0.158	0.365	–	–	–
Post Series A (=1)	111,780	0.102	0.303	–	–	–
Post Top-Tier VC (=1)	111,780	0.049	0.215	–	–	–
Post Mature Round (=1)	111,780	0.115	0.319	–	–	–

Note: The table reports summary statistics for the firm–month panel before stacking. SonarQube counts and remediation effort are summed across each firm's covered repositories, with gaps interpolated between observed months. Panels A–C use firm–months with a positive non-comment line count: most variables cover 61,768 firm–months, the Technical Debt Ratio covers 59,903 observations with nonmissing development effort, and headcount covers 32,313 firm–months. Panel A reports levels. Panel B divides issue counts by thousands of non-comment lines of code and Technical Debt by lines of code; the Technical Debt Ratio is calibrated debt divided by the replacement-cost proxy, multiplied by 100. Both debt measures use $\kappa = 68.60$ (Section 2.5). Panel C reports $\ln(1 + x)$ for each scale variable. Panel D reports indicators equal to one from the relevant financing month onward, over the full firm–month grid. Post Top-Tier VC uses a fixed list: Andreessen Horowitz, Lightspeed Venture Partners, Accel, Kleiner Perkins, Benchmark Capital Holdings, Greylock, Bessemer Venture Partners, and First Round Capital. Post Mature Round includes Series B and later venture rounds, acquisition financing, recapitalizations, corporate transactions, divestitures, buyouts, spin-offs, bonds, and private investments in public equity. These two descriptive indicators do not define the treatment used in our regressions.

treatment, a conventional two-way fixed-effects regression can compare newly funded firms with firms whose outcomes already reflect earlier financing. Under heterogeneous treatment effects, these comparisons can contaminate the estimated response and even reverse its sign (Callaway and Sant'Anna, 2021; Sun and Abraham, 2021). Following Gormley and Matsa (2011), we construct a separate comparison group for each financing cohort using firms that remain without a first round throughout its event window.

We group firms by the calendar month of their first completed Seed round, or their first completed Series A round when no Seed is recorded. For each cohort, controls are firms with no qualifying first round through the end of the window. The cohort panel covers 25 months, from event month $e = -12$ through $e = +12$ relative to financing. We then stack these panels.[7] After each sample restriction, we drop cohorts that contain only treated firms or only controls. The baseline contains 99 cohorts, 274 treated firms, 412 firms appearing as controls, and 484 unique firms. Appendix C describes the estimand and its relation to the pooled regression.

We estimate

$$M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}, \tag{7}$$

where i, c, and t index firm, cohort, and calendar month. $M_{i,c,t}$ is an engineering outcome, and $VC_{i,c,t}$ is $\text{Treated}_{i,c} \times \text{Post}_{c,t}$, which equals one for treated firms from the financing month onward. Cohort $\times$ Firm fixed effects, $\chi_{c,i}$, absorb persistent firm differences within each cohort; Month $\times$ Cohort fixed effects, $\delta_{c,t}$, absorb common monthly shocks within that cohort. $X_{i,c,t}$ denotes additional regressors in the heterogeneity specifications, and $\varepsilon_{i,c,t}$ is the error term. The baseline omits $X_{i,c,t}$.

The coefficient β captures the post-financing change in treated firms relative to their cohort controls. The pooled estimate weights these comparisons according to the observations and residual treatment variation in the stacked sample; Appendix C provides the formal interpretation. We cluster standard errors by cohort and firm to account for dependence within cohorts and across repeated appearances of the same firm.

We also examine two hypotheses about the scope of the improvement. Firms with more defects before financing may have greater room to improve, while firms raising larger rounds can devote more resources to engineers, tools, and remediation. The heterogeneity columns therefore interact treatment with *High Violations*, defined using critical-violation density at relative month -1, and *Large Round*, defined using the treated-firm median round size. These comparisons distinguish where the gains are concentrated and motivate the more direct mechanism tests below. Appendix F extends the analysis to founding-team characteristics using LinkedIn career histories, and Table 19 provides the complete variable definitions.

[7] A control firm–month can appear in several cohort panels, and a not-yet-treated firm can later enter as treated in its own cohort. The stacked observation count therefore exceeds the number of underlying firm–months.

To examine the timing of the response, we interact treated status with event-quarter indicators while retaining monthly fixed effects. We index the event-quarter bins from $q = -3$ to $q = 5$, omitting $q = 0$, and plot them at their month-bin midpoints. The omitted quarter contains relative months $\{-4, -3, -2\}$, and the quarter around financing, $q = 1$, contains $\{-1, 0, 1\}$. The latter bin includes one pre-financing month, so a change that begins only after the round closes is averaged with an untreated month. Quarters $q = 2$ through $q = 5$ contain only post-financing observations, with the final bin ending at month +12. These estimates compare pre-financing trajectories and trace how the improvement develops during the first year of institutional backing.

3.1 Identification Strategy

Venture capitalists select firms whose teams, products, and growth prospects they find promising. A central identification concern is therefore that VCs select firms whose engineering quality is already improving, or whose founders have planned improvements that have yet to appear in the code. Our identification does not require financing to be randomly assigned. It requires that, absent first VC entry, treated firms and their cohort controls would have followed parallel trends in engineering quality, together with no anticipatory response and no spillovers to controls.

Several features of our data and design address selection on an improving engineering trajectory. First, the event studies show little divergence in the main engineering-density outcomes before financing, followed by reductions around the first round. To explain this pattern, an omitted change in product demand, management, or engineering priorities would need to coincide with financing and produce the observed change in quality. A gradual improvement already under way before investor entry would instead predict a divergence in the pre-treatment coefficients.

Second, the timing of engineering quality differs from that of firm growth. Treated firms expand their headcount and codebases more rapidly than controls before financing, but their critical-violation and maintainability densities do not show a corresponding improvement. This distinction matters because VC selection on fast-growing firms need not imply selection on improving code quality. We develop this argument in the firm-scale results and test it directly by matching on pre-financing headcount growth in Section 4.1.

Third, we examine specific alternatives to a change in engineering practices. Commercial-stage and round-type tests ask whether the quality improvement is concentrated in firms making the initial transition to revenue generation. A Series B+ placebo asks whether the same quality gains recur when firms that already have institutional backing receive additional financing. Section 4 presents these tests and their implications for the interpretation of first VC entry.

Baseline Balance

Table 2 reports covariate means at the last pre-treatment month ($t = -1$). Treated and control firms enter the event with similar engineering quality: the standardized difference is 0.051 for critical violations per thousand lines of code, −0.025 for total violations, and −0.012 for both technical debt per line and the Technical Debt Ratio. The absence of a substantial baseline quality gap is informative for two competing selection stories. Treated firms neither start with markedly better code nor have systematically more defects to remove, as a simple room-for-improvement explanation would predict.

This balance also has an institutional rationale. Survey evidence indicates that VCs place particular weight on the management team and assess the product, technology, and market when selecting investments (Gompers et al., 2020). These assessments concern the commercial promise and capabilities of the venture; the detailed rule violations we measure capture a different margin, namely the implementation of coding practices within the existing product. An investor can therefore select a promising technology firm without selecting a codebase with low defect density. The observed balance is consistent with this distinction between commercial due diligence and the specific engineering outcomes in our analysis.

We formalize the baseline comparison with a linear probability model for treatment status at $t = -1$, including cohort fixed effects and clustering standard errors by firm. The three engineering-quality measures are not jointly predictive of financing ($F = 1.38$, $p = 0.25$). However, adding log codebase size leads the four-variable test to reject ($F(4, 19{,}928) = 3.86$, $p = 0.004$), with smaller codebases associated with treatment (-0.0016, $p < 0.01$). The within-cohort R^2 is 0.001. The selection detected in these observables concerns scale, while baseline engineering quality has little predictive power for financing conditional on timing.

The scale differences are economically substantial. Treated firms have lower total headcount (standardized difference −0.686), lower software-engineering headcount (−0.715), and smaller codebases (−0.308 in log lines of code). Cohort × Firm fixed effects absorb persistent differences in levels, and the matched specification addresses the different pre-financing growth trajectories. These comparisons support the design's focus on changes in quality around financing, while the causal interpretation ultimately requires that investors are not simply forecasting an unobserved engineering improvement that would have occurred at the same time without their entry.

Table 2: Baseline Balance: Treated vs. Control Firms at $t = -1$

	Treated	**Control**	**Std. Diff.**
Log(Lines of Code)	11.070	11.698	−0.308
Log(Headcount)	2.234	3.762	−0.686
Log(SWE Headcount)	1.465	2.883	−0.715
Critical Violations / kLoC	12.369	11.160	0.051
Technical Debt per LoC	12.453	12.675	−0.012
Violations / kLoC	30.769	31.861	−0.025
Technical Debt Ratio	45.283	46.091	−0.012
# Firm–cohort observations	247	19,686	

Note: This table reports covariate means at the last pre-treatment period ($t = -1$) for treated and control firms, using all control firms available in the stacked DiD panel. Firms can appear as controls in more than one cohort, so the observation counts refer to firm–cohort pairs. kLoC denotes 1,000 non-comment lines of code; SWE denotes software engineering. Log variables use $\log(1+x)$. Technical Debt per LoC is measured in dollars per line, and the Technical Debt Ratio is calibrated debt as a percentage of the codebase's replacement cost. Std. Diff. = (Treated Mean − Control Mean) / pooled standard deviation. See Appendix A, Table 19, for full variable definitions. Section 4.1 reports the matched-growth comparison.

3.2 Critical Code Defects

Critical violations and maintainability issues decline, while the bug and reliability estimates remain close to zero (Figure 1).

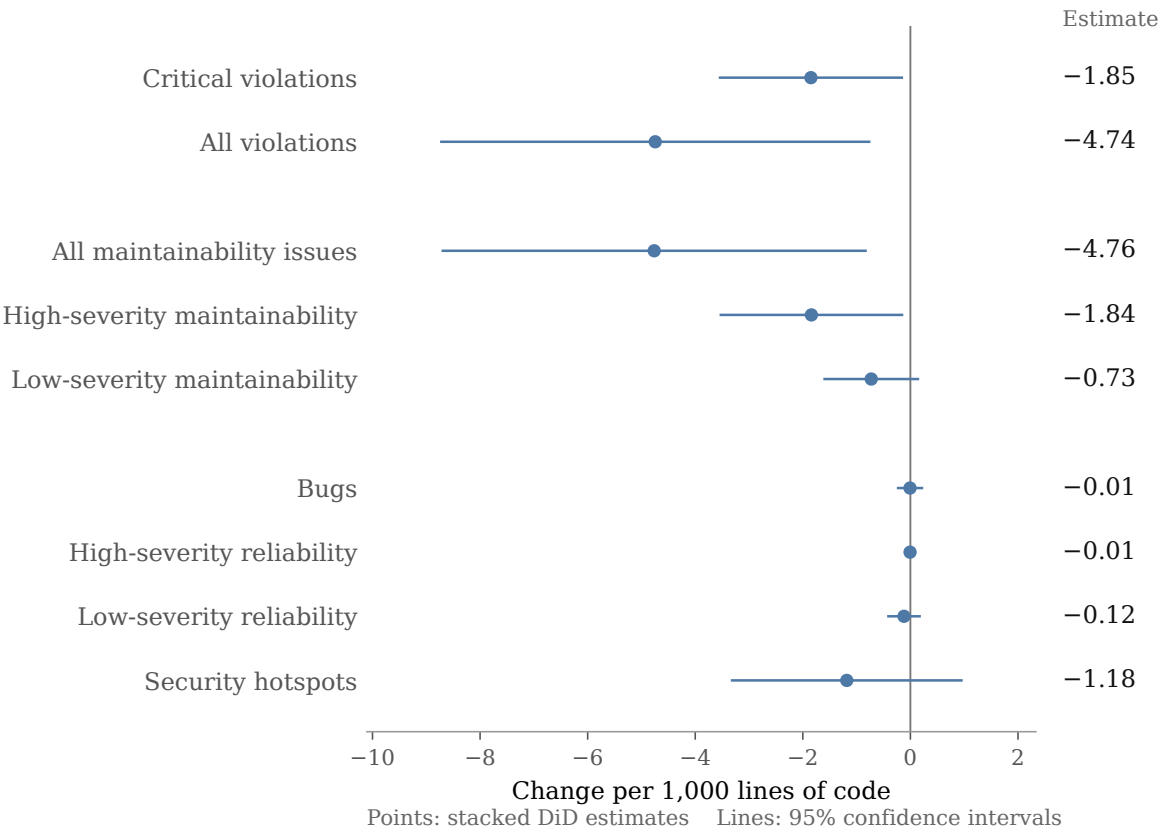


Figure 1: The Scope of the Engineering-Quality Improvement. Baseline estimates from Equation (7), in issues per 1,000 non-comment lines of code. Issue categories overlap. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

We first examine *Critical Violations*, severe deviations from coding best practices that can compromise the stability, security, or maintainability of a product. These issues receive high remediation priority in SonarQube's classification. Unlike a deliberate decision to postpone a minor improvement, allowing serious code problems to accumulate can make subsequent development both riskier and more costly. A decline in this category therefore speaks to the discipline with which firms manage consequential engineering problems.

Critical violations fall by 1.85 per 1,000 lines of code after financing relative to controls, approximately 15% of the treated pre-investment mean (Table 3, Column (1)). More broadly, total rule violations across all severity levels decline by 4.74 per 1,000 lines in Column (4), about 14% of the corresponding mean. Both estimates are significant at the 5% level.

Table 3: Difference-in-Differences Estimates for Critical Violations and All Violations

	Critical Violations			**Violations**		
	(1)	(2)	(3)	(4)	(5)	(6)
Treated × Post	−1.849**	−0.277	−2.290**	−4.744**	−4.600	−3.811**
	(0.863)	(0.390)	(1.145)	(2.016)	(2.980)	(1.720)
Post × High Violations (=1)		−1.306***			−2.174**	
		(0.317)			(0.998)	
Treated × Post × High Violations (=1)		−3.866**			−0.812	
		(1.892)			(3.740)	
Post × Large Round			0.017			−0.580
			(0.430)			(0.856)
Treated × Post × Large Round			1.000			−1.744
			(1.575)			(4.226)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	487,577	477,107	487,577	487,577	477,107	487,577
R^2	0.881	0.885	0.881	0.855	0.857	0.855

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Treated × Post indicator. Critical Violations are severe SonarQube rule violations per 1,000 lines of code. Violations (All) aggregate rule violations across all severity levels per 1,000 lines of code. See Appendix A, Table 19, for full variable definitions.

High Violations is a dummy equal to one if the firm's pre-treatment critical violation density (at $t = -1$) exceeds the median across baseline firm–cohort observations of firms treated somewhere in the sample, and zero otherwise.

Large Round is a dummy equal to one if the VC deal size (first Seed or Series A round amount, in USD) exceeds the treated-firm median, and zero otherwise.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

Figure 2 traces the timing of this improvement. The pre-investment estimates show no systematic divergence between treated and control firms, while negative coefficients emerge around financing and become larger over the post-investment quarters. The timing supports a shift in development practices following investor entry: it is not the pattern predicted by a gradual decline in critical-violation density already under way before the round. The fact that the improvement develops over subsequent quarters is also consistent with engineering practices taking time to affect the accumulated codebase.

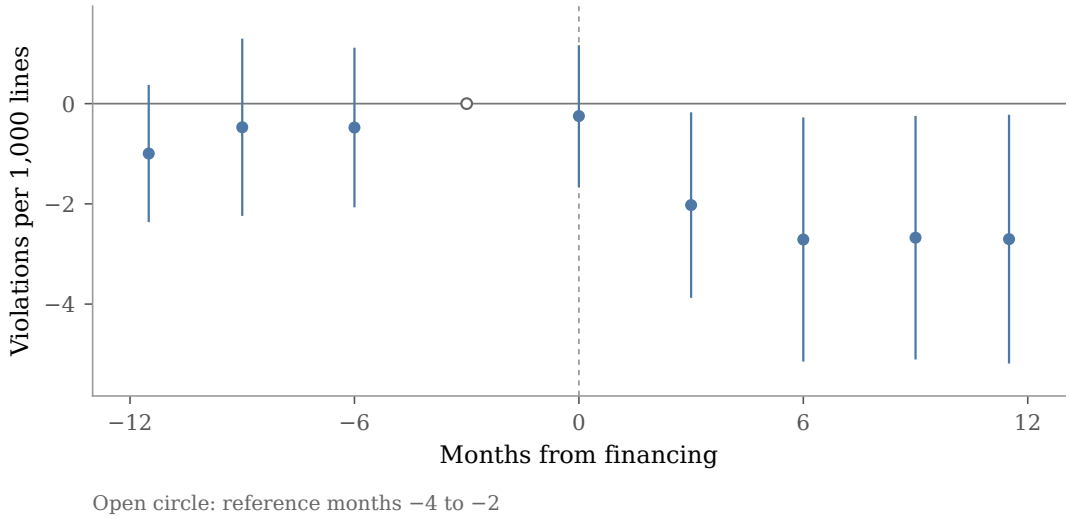


Figure 2: Critical Rule Violations per 1,000 Lines of Code. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

We observe a similar pattern for all rule violations in Figure 3. The post-financing coefficients are negative and generally increase in magnitude over the event window, indicating that the improvement extends beyond the Critical category and persists as firms continue to develop their software.

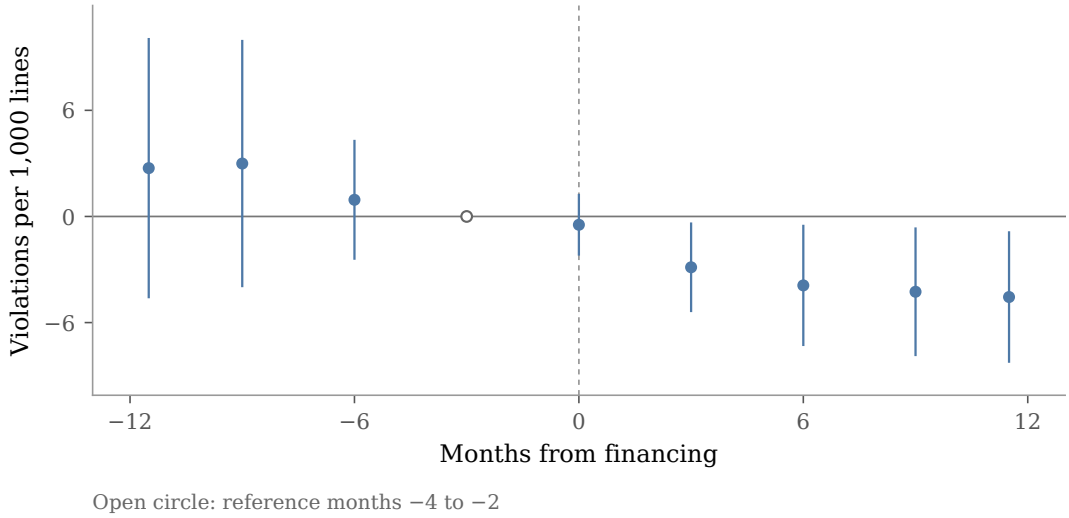


Figure 3: Rule Violations (All Severities) per 1,000 Lines of Code. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

3.3 Maintainability Issues and Security Hotspots

We next examine whether first VC entry improves the maintainability of software: the ease with which a codebase can be repaired, modified, or extended. Maintainability matters economically because an engineering shortcut that accelerates one release can increase the work required for every subsequent change. SonarQube flags code structures associated with this burden and classifies their severity according to their assessed impact. We study the overall measure and distinguish high-severity problems from lower-impact issues to determine whether the improvement reaches the more consequential parts of the codebase.

Maintainability issues decline by 4.76 per 1,000 lines of code after financing relative to controls, roughly 15% of the treated pre-investment mean and significant at the 5% level (Table 4, Column (1)). High-severity issues fall by 1.84 per 1,000 lines in Column (4), also approximately 15% of the pre-investment mean and significant at the 5% level. By contrast, the low-severity estimate in Column (7) is negative but statistically insignificant (−0.727, SE = 0.449). The improvement thus includes the code structures that make future development most costly.

Table 4: Difference-in-Differences Estimates for Maintainability Issues

	Maintainability Issues (All)			**Maintainability Issues (High)**			**Maintainability Issues (Low)**		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Treated × Post	−4.763**	−4.634	−3.872**	−1.840**	−0.271	−2.292**	−0.727	−0.683*	−1.025*
	(1.992)	(2.980)	(1.669)	(0.860)	(0.388)	(1.141)	(0.449)	(0.355)	(0.536)
Post × High Violations (=1)		−1.961**			−1.317***			−0.026	
		(0.960)			(0.317)			(0.511)	
Treated × Post × High Violations (=1)		−0.746			−3.859**			−0.140	
		(3.666)			(1.890)			(0.676)	
Post × Large Round			−0.693			0.010			−0.617*
			(0.823)			(0.429)			(0.350)
Treated × Post × Large Round			−1.571			1.030			1.106**
			(4.135)			(1.573)			(0.515)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	487,577	477,107	487,577	487,577	477,107	487,577	487,577	477,107	487,577
R^2	0.847	0.849	0.847	0.880	0.885	0.880	0.844	0.846	0.844

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Treated × Post indicator. Maintainability Issues (All) capture total maintainability-related rule violations per 1,000 lines of code—code patterns that hinder future modification or extension. High-Severity Maintainability Issues isolate the most severe problems indicating substantial technical debt accumulation. Low-Severity Maintainability Issues capture minor structural or readability deficiencies. See Appendix A, Table 19, for full variable definitions. See Table 3 for the definitions of the High Violations and Large Round dummies.
Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

The event studies in Figures 4 and 5 reinforce these findings. Pre-treatment coefficients show no systematic divergence in either outcome, and maintainability issues decline around financing.

The timing mirrors the reduction in critical violations, consistent with a broader improvement in engineering practices.

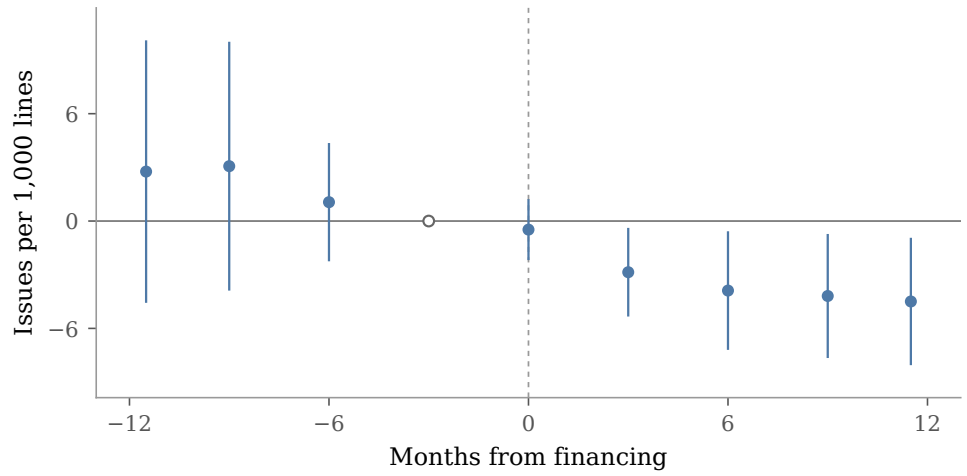


Figure 4: Maintainability Issues per 1,000 Lines of Code. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

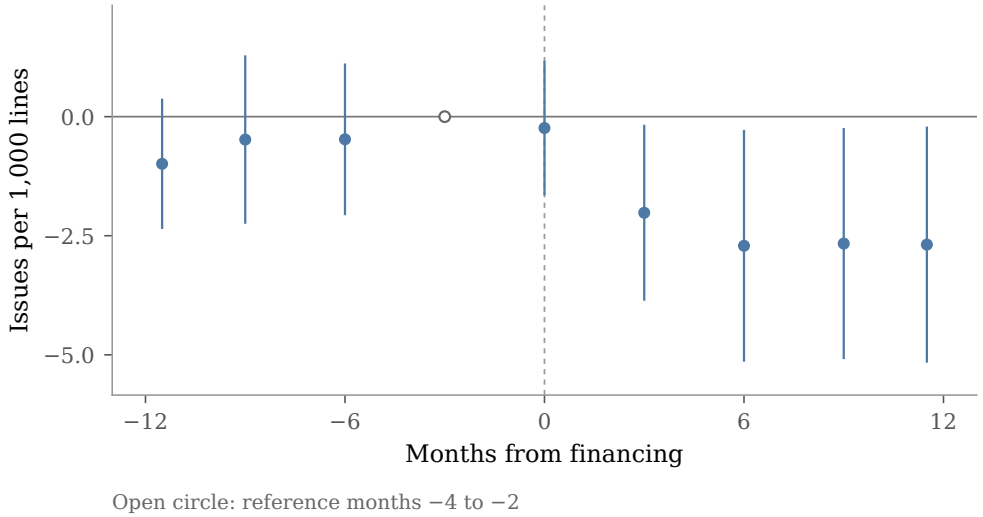


Figure 5: Maintainability Issues (High-Severity) per 1,000 Lines of Code. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

We also examine *Security Hotspots*.[8] Security Hotspots identify security-sensitive code seg-

[8] We omit the "Open Issues" metric from the analysis. In retrospective scanning—where the full codebase is analyzed at each snapshot rather than through a live triage workflow—every rule violation is simultaneously an open issue. The two variables are therefore identical in our data, and reporting both would double-count the same information.

ments that require human review to determine whether a vulnerability exists. Table 5 reports the estimates.

The baseline estimate in Column (1) is negative but statistically insignificant (−1.183 hotspots per 1,000 lines of code, SE = 1.086). The large-round interaction is also insignificant. We therefore find no systematic change in security hotspots after first VC entry.

Table 5: Difference-in-Differences Estimates for Security Hotspots

	Security Hotspots		
	(1)	(2)	(3)
Treated × Post	−1.183	−2.083	−0.022
	(1.086)	(1.931)	(0.091)
Post × High Violations (=1)		0.105	
		(0.088)	
Treated × Post × High Violations (=1)		2.050	
		(1.921)	
Post × Large Round			−0.100
			(0.112)
Treated × Post × Large Round			−2.595
			(2.382)
Cohort × Firm FE	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes
# Obs	487,577	477,107	487,577
R^2	0.725	0.704	0.725

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Treated × Post indicator. Security Hotspots measure security-sensitive code segments requiring manual audit per 1,000 lines of code; unlike rule violations, hotspots represent potential risk areas requiring human assessment. See Appendix A, Table 19, for full variable definitions.

See Table 3 for the definitions of the High Violations and Large Round dummies.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

3.4 Bugs and Reliability

We next examine bugs and reliability issues, which concern incorrect behavior or failures during software execution. SonarQube's Bugs measure identifies code patterns that are likely to produce an incorrect result or unexpected behavior. The reliability measures capture related risks using the newer classification, with high-severity issues carrying greater potential impact and low-severity issues carrying smaller risks. Both assess problems identifiable from source code, allowing us to examine whether the gains in maintainability extend to the software's expected behavior in use.

We find no statistically significant improvement in bugs or reliability (Table 6). The Bugs coefficient in Column (1) is −0.007 per 1,000 lines of code (SE = 0.123), small relative to the treated pre-investment mean of 1.82. The high- and low-severity reliability estimates in Columns (4) and (7) are also negative and statistically insignificant. Figure 6 presents the corresponding event studies. Together with the security-hotspot results, these findings indicate that the measurable gains are concentrated in rule compliance and maintainability. Firms face a lower burden of modifying and extending the product, while we detect no comparable change in the code patterns associated with software failures.

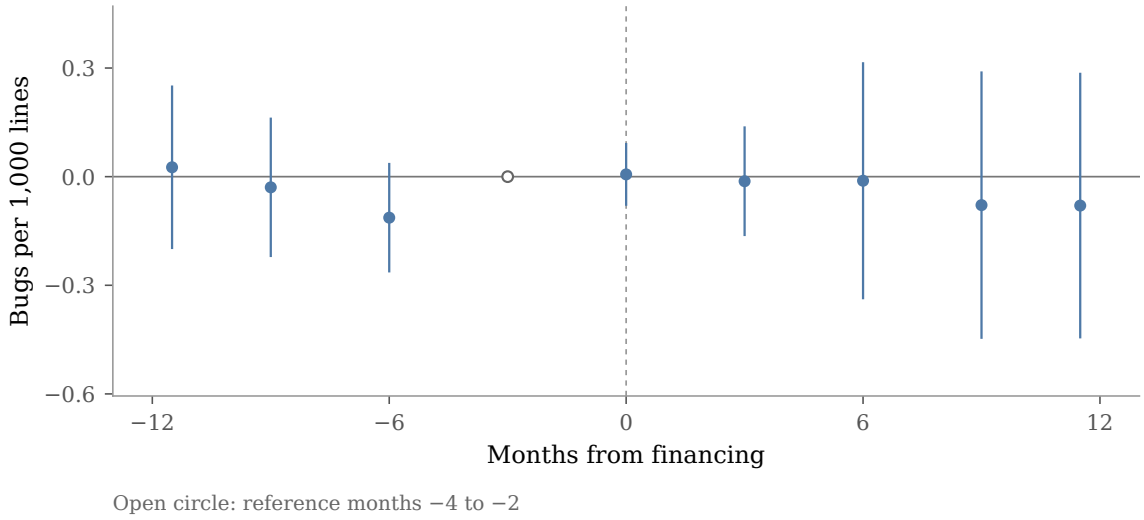
Table 6: Difference-in-Differences Estimates for Bugs and Reliability Issues

	Bugs			**Reliability Issues (High)**			**Reliability Issues (Low)**		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Treated × Post	−0.007	−0.013	0.042	−0.006	0.002	0.002	−0.119	−0.312	−0.182
	(0.123)	(0.085)	(0.148)	(0.016)	(0.013)	(0.023)	(0.158)	(0.251)	(0.217)
Post × High Violations (=1)		−0.225**			0.011			−0.080	
		(0.110)			(0.016)			(0.103)	
Treated × Post × High Violations (=1)		−0.020			−0.019			0.428	
		(0.273)			(0.030)			(0.280)	
Post × Large Round			0.097			0.006			−0.031
			(0.082)			(0.008)			(0.068)
Treated × Post × Large Round			−0.179			−0.022			0.165
			(0.246)			(0.025)			(0.220)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	487,577	477,107	487,577	487,577	477,107	487,577	487,577	477,107	487,577
R^2	0.903	0.904	0.903	0.505	0.507	0.505	0.903	0.904	0.903

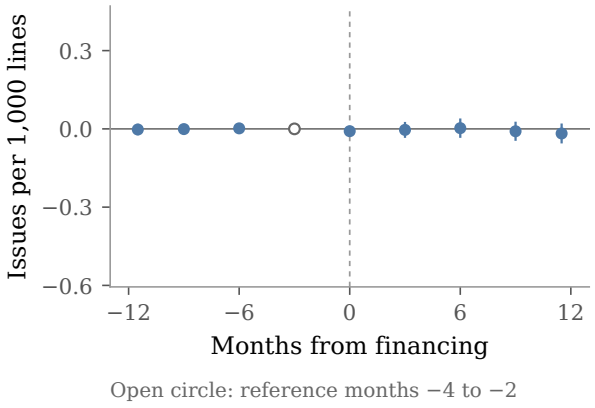
This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Treated × Post indicator. Bugs are patterns classified by SonarQube as likely to cause incorrect or unexpected behavior per 1,000 lines of code. High- and Low-Severity Reliability Issues capture stability-related rule violations at the stated severity. See Appendix A, Table 19, for full variable definitions.

See Table 3 for the definitions of the High Violations and Large Round dummies.

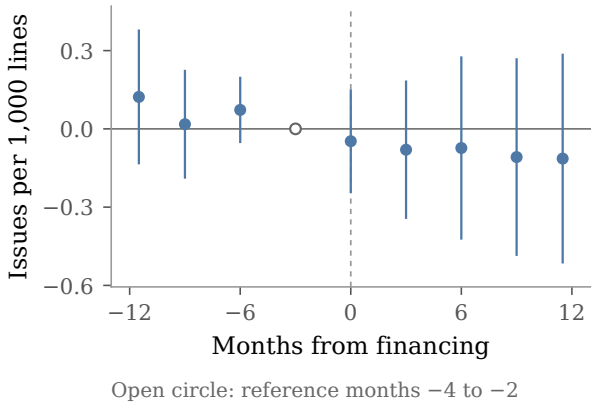
Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.



(a) Bugs per 1,000 Lines of Code



(b) Reliability Issues (High Severity) per 1,000 lines of code



(c) Reliability Issues (Low Severity) per 1,000 lines of code

Figure 6: Bugs and Reliability Issues. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

3.5 Technical Debt

Better coding discipline can reduce the future cost of maintaining software. We measure this liability with *Technical Debt*, which converts SonarQube's model-implied remediation time into dollars using the calibration in Section 2.5. The *Technical Debt Ratio* expresses the same liability as a percentage of the software asset's estimated replacement cost. Both measures translate engineering problems into an estimated maintenance burden.

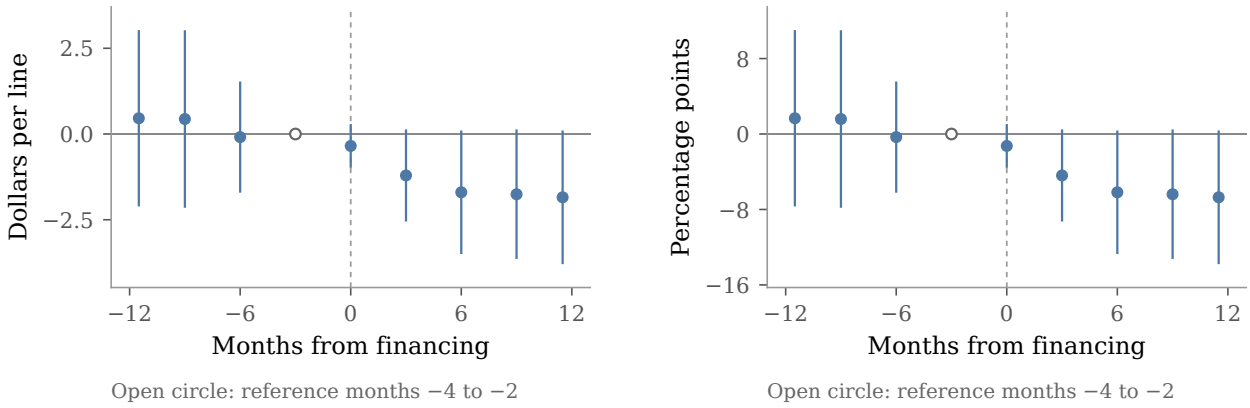
Table 7, Column (1), shows that technical debt declines by $1.57 per line of code after financing

relative to controls, approximately 12% of the treated pre-investment mean. Correspondingly, the Technical Debt Ratio falls by 5.69 percentage points in Column (4), also about 12% of its 47.6% pre-investment mean. Both estimates are significant at the 5% level. Under the common calibration, this ratio decline means that the estimated maintenance liability falls by $5.69 for every $100 of software replacement cost. The improvement therefore represents a meaningful reduction in the future work associated with maintaining a given amount of software, in addition to the decline in unweighted issue counts. Figure 7 shows how the two measures evolve around financing.

Table 7: Difference-in-Differences Estimates for Technical Debt

	Technical Debt per LoC ($)			**Technical Debt Ratio**		
	(1)	(2)	(3)	(4)	(5)	(6)
Treated × Post	−1.566**	−1.453	−1.248	−5.693**	−5.285	−4.539
	(0.765)	(0.979)	(0.813)	(2.782)	(3.559)	(2.956)
Post × High Violations (=1)		−0.525			−1.907	
		(0.516)			(1.878)	
Treated × Post × High Violations (=1)		−0.400			−1.454	
		(1.458)			(5.302)	
Post × Large Round			−0.194			−0.706
			(0.363)			(1.318)
Treated × Post × Large Round			−0.595			−2.165
			(1.510)			(5.490)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	487,577	477,107	487,577	487,577	477,107	487,577
R^2	0.828	0.828	0.828	0.828	0.828	0.828

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Treated × Post indicator. Technical Debt per Line of Code is the calibrated dollar remediation burden per line. The Technical Debt Ratio is calibrated technical debt as a percentage of the firm's software assets, the replacement cost of its codebase. Section 2.5 defines both outcomes.
See Table 3 for the definitions of the High Violations and Large Round dummies.
Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.



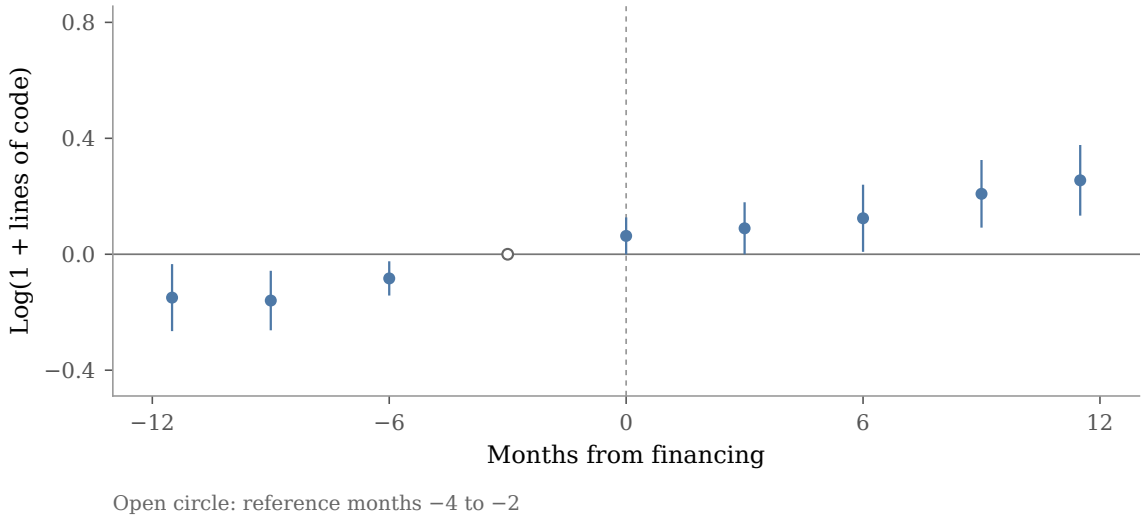
(a) Technical Debt per Line of Code ($)

(b) Technical Debt Ratio

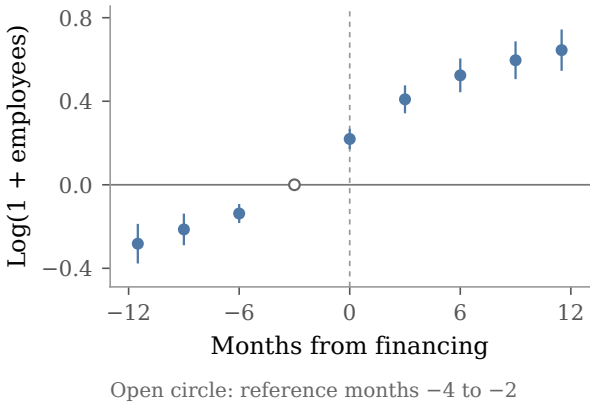
Figure 7: Technical Debt Measures. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

3.6 Firm Scale and Engineering Capacity

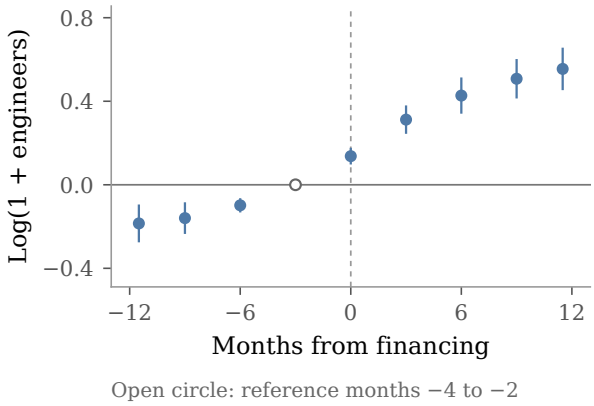
We next examine the expansion in firm scale and engineering capacity that accompanies first VC entry. This expansion provides the setting in which the quality improvements occur: firms are adding employees and extending their products while reducing the density of engineering defects. Figure 8 plots the dynamic estimates for code volume in Panel (a), total headcount in Panel (b), and software-engineering headcount in Panel (c), with each outcome transformed as $\log(1 + x)$.



(a) Log(Lines of Code)



(b) Log(Total Employees)



(c) Log(Software Engineers)

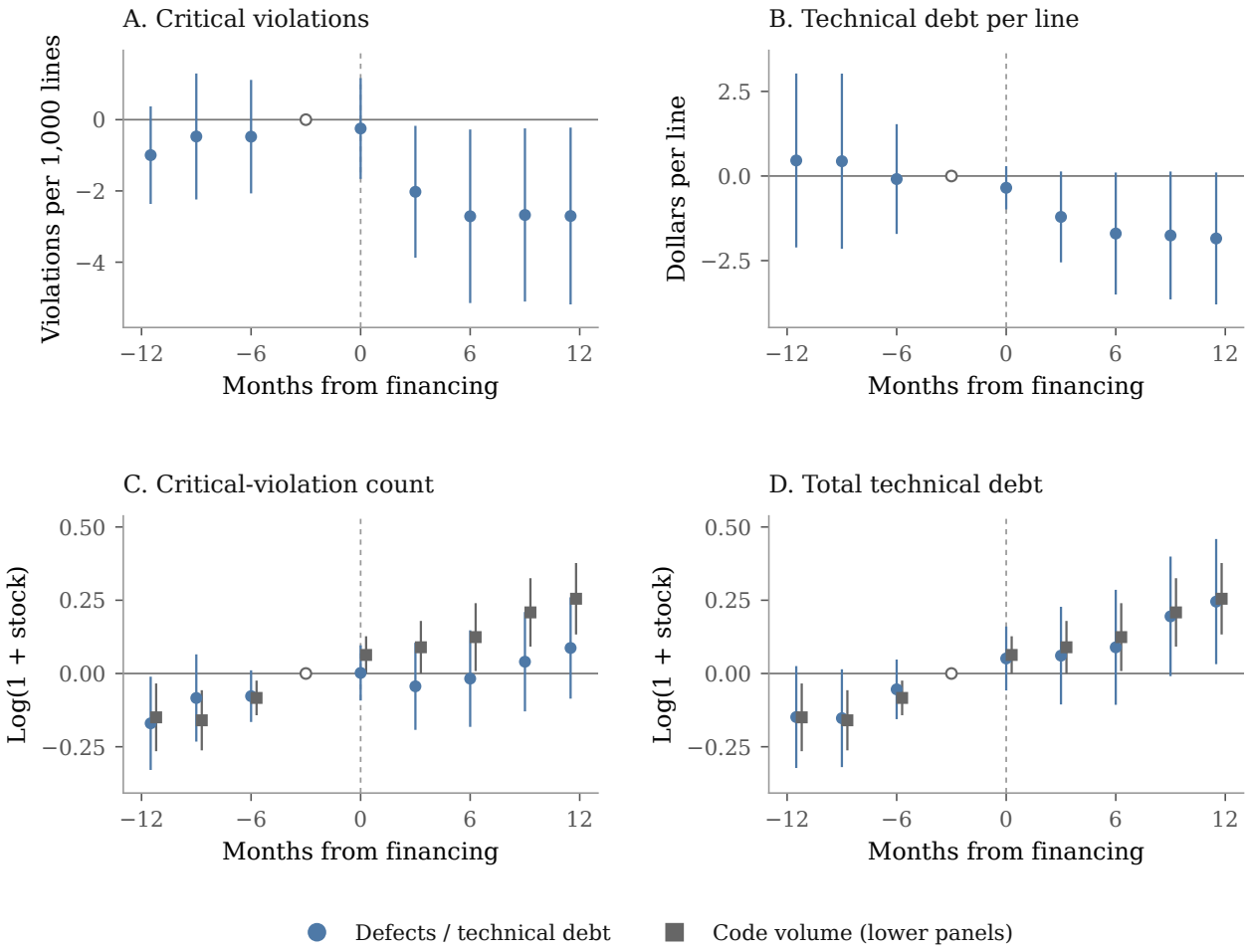
Figure 8: Firm and Product Scaling. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

Organizational scale rises substantially after financing. By the end of the event window, total headcount is approximately 65 log points higher relative to controls, and software-engineering headcount is approximately 55 log points higher. Code volume also expands, although by less than headcount. This gap is consistent with the demands of building a larger engineering organization—including onboarding, review, and coordination—that need not translate immediately into proportionate additions to the codebase. It also cautions against treating the number of lines produced as a sufficient measure of engineering output.

The pre-treatment coefficients show that treated firms were already expanding more rapidly in all three scale measures before financing. Parallel trends therefore do not hold for these outcomes,

consistent with VC selection on fast-growing firms. The scaling estimates describe the resource expansion surrounding investor entry; the identification of our quality results depends on the trajectories of the engineering outcomes themselves.

Figure 9 shows how quality changes as the codebase expands. The upper panels report defect and debt densities; the lower panels compare their stocks with code volume. Appendix G discusses the relationship between these measures.



Open circle: reference months −4 to −2. Bars: 95% pointwise confidence intervals.

Figure 9: Engineering Quality During Codebase Expansion. Panels A–B show critical violations per 1,000 lines and technical debt in dollars per line. Panels C–D compare defect and debt stocks with code volume, using $\log(1 + x)$. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

Growth and quality follow different trajectories before financing. If lower defect density continued the same gradual improvement in firm capabilities that supported earlier growth, we would expect quality to improve before the round as well. Headcount and code volume diverge before fi-

nancing, while the decline in critical violations and maintainability issues emerges around investor entry. This separation in timing supports a change in development practices beyond the continuation of pre-existing growth. We test that explanation directly by matching treated firms to controls with similar pre-financing headcount growth in Section 4.1.

Rapid expansion also creates engineering challenges. More developers must coordinate changes to an increasingly complex product, and onboarding new engineers places demands on the existing team. Quality need not improve automatically as resources increase. The decline in defect density during this expansion is consistent with firms adopting practices that help them manage a larger development effort.

An alternative explanation is that investors improve quality by recruiting senior technical talent who write or supervise better code. Section 3.7 examines this possibility using LinkedIn employment histories. The gains are not concentrated in firms that recruit a technical leader or a larger share of senior engineers, and the process evidence in Section 3.8 shows improvements in contributions made by engineers already present before financing. Together, these findings direct attention to changes in how the existing organization develops software.

3.7 Capital and Governance: Mechanism Evidence

The results thus far show that first institutional VC entry is followed by lower defect density as firms expand. We next examine whether these improvements reflect the resources financing provides, the recruitment of senior technical talent, or changes in governance and development practices. The explanations imply different roles for investors. Additional capital can finance engineers, better tools, and technical-debt remediation; if these resources account for the gains, the improvement need not depend on how investors organize oversight. A governance channel instead predicts that quality responds to the monitoring and professionalization accompanying investment—including board oversight, operating advice, and more formal development procedures. We examine these predictions using investor structure and the composition of post-financing hiring.

We first exploit variation in investor count. In larger syndicates, each investor bears the private cost of monitoring while sharing its benefits with more co-investors (Lerner, 1994; Casamatta and Haritchabalet, 2007; Kang, Li, and Oh, 2022). If these dispersed incentives weaken oversight, firms financed by larger syndicates should experience smaller quality improvements. To test this prediction, we classify a round as a large syndicate if it exceeds the treated-firm 75th percentile of eight investors. This yields 60 treated firms with large syndicates among the 260 with nonmissing counts. We also report estimates using the median cutoff of four investors. We interact these indicators with treatment and estimate:

$$M_{i,c,t} = \beta_1 VC_{i,c,t} + \beta_2 VC_{i,c,t} \times \text{Syndicate}_i + \beta_3 \text{Post}_{c,t} \times \text{Syndicate}_i + \chi_{c,i} + \delta_{c,t} + \varepsilon_{i,c,t}, \quad (8)$$

where β_1 is the estimate for the reference group and $\beta_1 + \beta_2$ is the estimate for rounds with large syndicates. A positive β_2 indicates smaller quality improvements when more investors share the round, as predicted by weaker individual monitoring incentives.[9]

Table 8 reports the predicted attenuation in critical violations under both syndicate cutoffs. At the 75th-percentile cutoff in Column (3), the interaction is +1.581 (SE = 1.618) and statistically insignificant. At the median cutoff in Column (4), it is +3.002 (SE = 1.666), significant at the 10% level, leaving a net point estimate close to zero for larger syndicates. Thus, firms financed by fewer co-investors experience the larger estimated reduction in critical violations. However, the technical-debt and code-volume interactions are insignificant at both cutoffs, so the statistical support for this monitoring prediction is concentrated in critical violations under the median definition.

Table 8: Investor Structure and Engineering Outcomes

	Critical Violations / kLoC				**TD per LoC ($)**				**Log(Lines of Code)**			
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
Treated × Post	−1.849**	−1.867**	−2.197**	−3.110**	−1.566**	−1.591**	−1.747*	−2.251*	0.226***	0.226***	0.187***	0.215***
	(0.863)	(0.872)	(1.045)	(1.423)	(0.765)	(0.773)	(0.909)	(1.219)	(0.048)	(0.049)	(0.054)	(0.064)
Post × Angel		−0.205				−0.232				0.006		
		(0.399)				(0.378)				(0.087)		
Treated × Post × Angel		1.707				2.457**				0.005		
		(1.815)				(1.111)				(0.238)		
Post × Syndicate p75			0.204				0.341				0.135	
			(0.809)				(0.507)				(0.088)	
Treated × Post × Syndicate p75			1.581				0.634				0.083	
			(1.618)				(1.214)				(0.127)	
Post × Syndicate p50				−0.086				−0.197				0.163***
				(0.475)				(0.427)				(0.053)
Treated × Post × Syndicate p50				3.002*				1.742				−0.096
				(1.666)				(1.376)				(0.109)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	487,577	487,577	487,577	487,577	487,577	487,577	487,577	487,577	487,577	487,577	487,577	487,577
R^2	0.881	0.881	0.881	0.881	0.828	0.828	0.828	0.828	0.931	0.931	0.931	0.931

This table reports heterogeneity in the treatment effect by investor type. Angel is an indicator equal to one if the firm's first qualifying capital round (Angel, Seed, or Series A) is an Angel (individual) round (4 of 274 treated firms). Syndicate p75 and p50 are indicators equal to one if the first Seed or Series A treatment round has more investors than the treated-firm 75th percentile (8) or median (4), respectively. These definitions classify 60 and 120 treated firms as syndicated among the 260 treated firms with non-missing investor counts. The 14 treated firms with missing counts remain in the reference group. Code volume is transformed as log(1 + LoC). Columns (1), (5), and (9) reproduce the baseline estimates. Columns (2), (6), and (10) introduce the angel interaction. Columns (3), (7), and (11) introduce the p75 syndicate interaction; Columns (4), (8), and (12) use the p50 syndicate interaction. All specifications include Firm × Cohort and Month × Cohort fixed effects. See Appendix A, Table 19, for variable definitions.

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

Columns (2), (6), and (10) provide a complementary comparison based on prior angel financing.

[9]The reference group includes firms at or below the cutoff and firms with missing investor counts, which the specification codes as zero.

Institutional VC contracts commonly allocate board seats and formal control rights to investors, whereas angel financing can rely more heavily on informal relationships and advice (Kaplan and Strömberg, 2003; Ibrahim, 2008; Chemmanur and Chen, 2014). This difference motivates examining firms whose first qualifying capital round was an Angel (individual) round, while retaining first Seed or Series A entry as the treatment event. Four of the 274 treated firms meet this definition. Their net critical-violation estimate is −0.160, and the technical-debt interaction is +2.457 (SE = 1.111, $p < 0.05$), indicating smaller gains for these firms at subsequent institutional entry.

To examine quality at angel entry itself, we estimate a separate DiD centered on angel financing, censoring treated firms when they subsequently receive institutional funding. None of the quality coefficients is statistically significant, and all four point estimates are positive: critical violations +0.13, total violations +2.26, maintainability issues +2.33, and technical debt +0.44 per line. These estimates provide no evidence of quality gains at angel entry. Only five angel-financed firms have sufficient codebase coverage, however, forming five cohorts against 461 controls (31,678 stacked observations); we therefore treat the angel comparison as supporting evidence rather than a central test of the mechanism.

Senior Technical Hiring

An alternative explanation is that VCs improve code quality by facilitating the hiring of senior technical talent. We examine this possibility using two measures constructed from LinkedIn employment histories. *Hired Technical Leader* identifies a CTO, VP of Engineering, or Head of Engineering hire within 24 months of financing. *High Senior Ratio* identifies firms whose senior engineering hires as a share of all new hires over that window exceed the treated-firm median. We restrict these comparisons to firms with LinkedIn coverage and estimate:

$$M_{i,c,t} = \beta_1 VC_{i,c,t} + \beta_2 VC_{i,c,t} \times H_i + \beta_3 \text{Post}_{c,t} \times H_i + \chi_{c,i} + \delta_{c,t} + \varepsilon_{i,c,t}, \tag{9}$$

where H_i denotes the hiring-composition indicator. If senior technical recruitment accounts for the quality gains, firms that add this talent should improve more, producing a negative interaction. The comparison tests whether the observed pattern fits this prediction; the hiring measures themselves are recorded after financing.

Table 9 shows that quality gains are not concentrated in firms recruiting senior technical talent. Both critical-violation interactions are positive: +4.674 (SE = 3.938) for a hired technical leader and +2.025 (SE = 2.981) for a high senior-hire ratio, although neither is statistically significant. For total violations and maintainability, the hired-leader interactions are positive and significant at the 10% level (+10.411, SE = 5.370, and +9.255, SE = 5.013). In these outcomes, firms adding a technical leader experience smaller estimated improvements. Firms without a recorded leader hire retain negative base

estimates of −9.624 ($p < 0.05$) and −8.751 ($p < 0.10$), respectively. The improvement therefore reaches firms that do not recruit a new technical leader, within the LinkedIn-covered sample of 68,498 observations.

Table 9: Engineering Quality by Post-Financing Hiring Composition

	Critical Violations / kLoC			**Violations / kLoC**			**TD per LoC ($)**			**Maintainability Issues / kLoC**		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
Treated × Post	−1.849**	−5.123	−3.354	−4.744**	−9.624**	−5.286	−1.566**	−3.134	−1.862	−4.763**	−8.751*	−4.846
	(0.863)	(3.918)	(3.008)	(2.016)	(4.774)	(3.761)	(0.765)	(2.217)	(1.683)	(1.992)	(4.530)	(3.580)
Post × Hired Technical Leader		−0.894			−3.859			−1.122			−3.192	
		(1.089)			(2.678)			(1.273)			(2.398)	
Treated × Post × Hired Technical Leader		4.674			10.411*			3.319			9.255*	
		(3.938)			(5.370)			(2.577)			(5.013)	
Post × High Senior Ratio			0.530			−0.222			−0.291			0.140
			(0.956)			(2.007)			(0.877)			(1.865)
Treated × Post × High Senior Ratio			2.025			3.423			1.391			2.949
			(2.981)			(4.292)			(1.983)			(4.044)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	487,577	68,498	68,498	487,577	68,498	68,498	487,577	68,498	68,498	487,577	68,498	68,498
R^2	0.881	0.815	0.815	0.855	0.828	0.827	0.828	0.783	0.783	0.847	0.816	0.816

This table reports heterogeneity in the treatment effect by post-funding hiring composition. Hired Technical Leader is an indicator equal to one if the firm hired a CTO, VP of Engineering, or Head of Engineering within 24 months of the funding event. High Senior Ratio is an indicator equal to one if senior engineering hires as a share of all new hires over that window exceed the treated-firm median. Columns (1), (4), (7), and (10) reproduce the baseline estimates on the full sample. Columns (2), (5), (8), and (11) introduce the hired-technical-leader interaction; Columns (3), (6), (9), and (12) introduce the high-senior-ratio interaction. The interaction columns use the subsample with LinkedIn coverage; firms without coverage are excluded. Hiring composition is measured after financing, so these interactions are descriptive comparisons across post-treatment groups. All specifications include Firm × Cohort and Month × Cohort fixed effects. See Appendix A, Table 19, for variable definitions.

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

We next examine hiring composition as an outcome. The technical- and nontechnical-leadership indicators measure whether the firm hires a leader in each of two periods: months $[-12, -2]$ and $[-1, 12]$ relative to financing. The software-engineering share and mean seniority of new hires are measured quarterly. Table 10 reports the corresponding DiD estimates, with cohort-by-firm fixed effects and either a period effect or cohort-by-quarter effects.

The software-engineering share of new hires rises by 5.8 percentage points (SE = 2.9 percentage points), significant at the 5% level. By contrast, we find no statistically significant change in technical or nontechnical leadership hiring or in the mean seniority of engineering hires. Financing shifts recruitment toward engineering, with no detectable shift toward more senior technical personnel. Combined with the interaction estimates, these results offer little support for senior recruitment as the principal explanation for the quality improvement.

Table 10: Hiring Composition: Direct Outcome-Based DiD

	Pre/Post 2×2		Quarterly Panel	
	Tech. Leader	Non-tech. Leader	SWE Share	SWE Seniority
	(1)	(2)	(3)	(4)
Treated × Post	0.055	−0.036	0.058**	−0.109
	(0.048)	(0.056)	(0.029)	(0.097)
Cohort × Firm FE	Yes	Yes	Yes	Yes
Post FE	Yes	Yes	—	—
Quarter × Cohort FE	—	—	Yes	Yes
# Obs	12,710	12,710	31,339	20,889
# Firms	258	258	260	217
R^2	0.748	0.744	0.416	0.488

This table reports the coefficient on Treated × Post from a direct outcome-based DiD specification for four hiring-composition outcomes constructed from LinkedIn position histories. Columns (1) and (2) collapse each (firm, cohort) panel to two period rows—pre = relative month $\in [-12, -2]$ and post = relative month $\in [-1, 12]$—and estimate $Y_{i,c,p} = \beta \cdot VC_{i,c,p} + \chi_{c,i} + \delta_p + \varepsilon_{i,c,p}$. $Y_{i,c,p}$ is the corresponding leadership-hiring indicator. *Tech. Leader* is an indicator equal to one if the firm hired a CTO, VP of Engineering, Head/Director of Engineering, or VP of Technology during the period. *Non-tech. Leader* is the analogous indicator for a nontechnical executive (CEO, COO, CFO, CMO, CPO, CRO, CCO, or a head/VP/director of a non-engineering function).

Columns (3) and (4) estimate the canonical quarterly stacked DiD, $M_{i,c,t} = \beta \cdot VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \varepsilon_{i,c,t}$, at firm–cohort-quarter frequency. *SWE Share* is the share of distinct new hires in the quarter mapped to a software-engineering individual-contributor role. *SWE Seniority* is the mean LinkedIn seniority (1–7 scale) of those hires. Both ratios are missing when the denominator is zero, and firms without LinkedIn coverage drop from all four columns rather than being coded as zero. See Appendix A, Table 19, for full variable definitions.

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

Mechanism Synthesis

The investor and hiring comparisons help distinguish three explanations for the quality gains. First, the syndicate gradient is consistent with monitoring incentives: firms financed by fewer co-investors experience larger reductions in critical violations, with marginal statistical support under the median cutoff. Second, the angel-centered estimates do not reproduce the institutional-entry gains, although their small sample makes them corroborating evidence. Third, the hiring tests provide no evidence that improvements are concentrated in firms adding senior technical talent, and substantial gains remain among firms without a new technical leader. This combination supports a role for governance and development practices beyond the simple addition of capital or senior personnel.

These comparisons use investor and hiring characteristics as proxies for the channels at work. Syndicate size can also capture differences in expertise and financing structure, and firms may recruit

technical leaders in response to engineering problems. We therefore interpret the pattern jointly, rather than assigning a causal contribution to each component of the financing relationship. If professionalization reaches the existing organization, we should observe changes in how its engineers produce software. Section 3.8 examines this prediction using individual development records, connecting our findings to the evidence on VC involvement in portfolio-company operations (Gorman and Sahlman, 1989; Hellmann and Puri, 2002; Bernstein, Giroud, and Townsend, 2016).

3.8 Changes in the Development Process

In order to shed more light into what happens inside the firm after VC wntry, we examine changes in development practices using 4.8 million individual code contributions. Version-control histories record when engineers submit changes, how they document and integrate them, and who last modified each line of code. We use these records to test whether more formal procedures and better output from the existing team accompany the improvement in engineering quality. The analysis uses a separate sample of firms with one repository, independent of the SonarQube scans. Appendix M describes sample coverage and estimation details.

First, teams adopt more formal procedures for reviewing and coordinating development. Table 11 shows that the share of commits carrying a review or sign-off marker rises by 3.3 percentage points, significant at the 1% level. Teams also link more changes to numbered tasks or bugs and to pull requests—documented proposals that collect changes for review before integration. The shares referencing a tracked task, linking to a pull request, and recording a merge or squash each rise by roughly 6 percentage points, significant at the 5% level or better.[10] Teams therefore record and coordinate development more systematically after investor entry.

Figure 10 compares the changes in documented practices, work timing, and the share of incumbent contributions linked to later bug fixes.

[10] A merge combines development histories; a squash incorporates a set of changes as one commit. A sign-off can certify a contribution without independent review, so the review measure captures documented review or certification procedures.

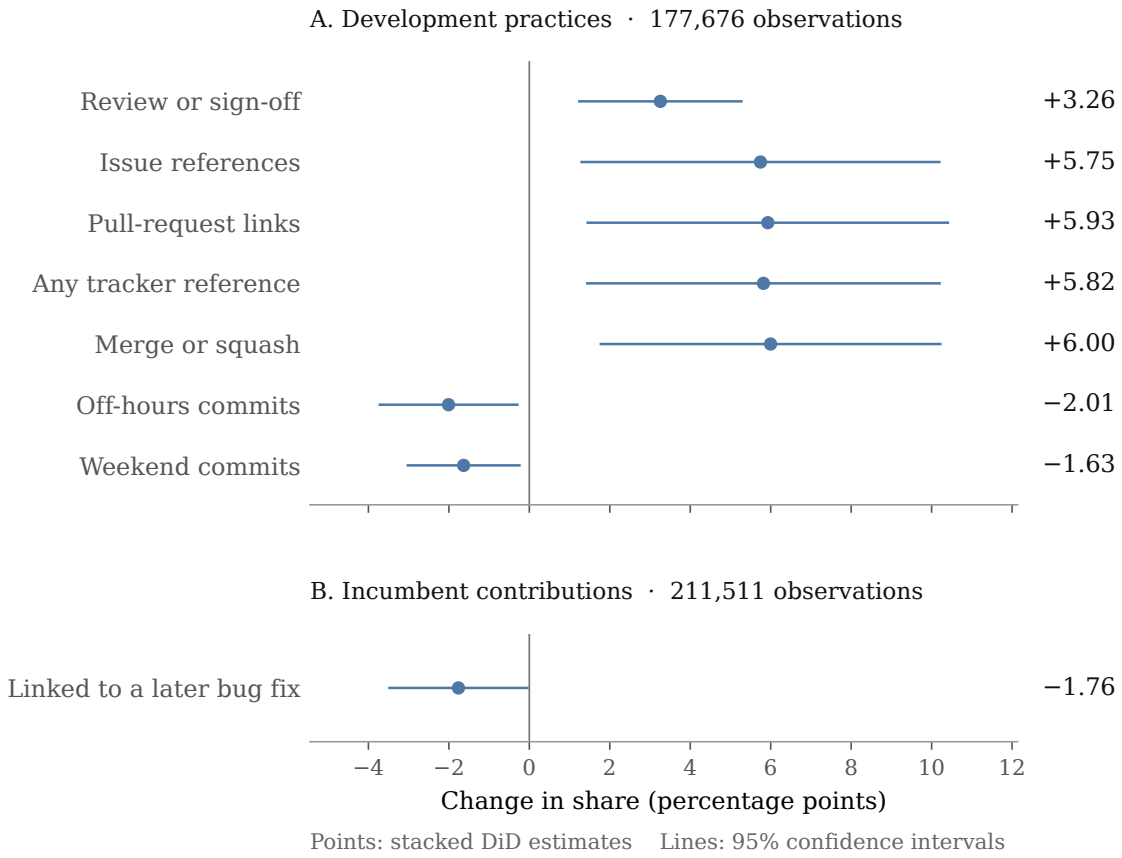


Figure 10: Development Practices and Contributions by Incumbent Engineers. Changes in monthly commit shares, in percentage points. Panel A shows documentation and timing measures; Panel B shows commits by pre-financing engineers linked to later bug fixes. The panels use separate single-repository samples. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

Table 11: Engineering Process Measures After First VC Entry

Practice	Dependent Variable	Treated × Post	p-value
Code review	Review or sign-off trailer share	0.0326***	0.002
		(0.0103)	
Issue tracking	Issue-reference share	0.0575**	0.013
		(0.0225)	
Pull-request workflow	PR-link share	0.0593**	0.011
		(0.0227)	
Formal tracking	Any tracker-reference share	0.0582**	0.010
		(0.0222)	
PR integration	Merge or squash share	0.0600***	0.006
		(0.0214)	
Working hours	Off-hours commit share	−0.0201**	0.024
		(0.0087)	
Working hours	Weekend commit share	−0.0163**	0.025
		(0.0071)	
Cohort × Firm FE	Yes		
Month × Cohort FE	Yes		
# Obs	177,676		

Each row estimates Equation (7) on the single-repository panel (212 treated firms). Each share is the fraction of a firm's commits in a month carrying the stated marker. The review or sign-off share records trailer lines such as "Reviewed-by:" or "Signed-off-by:"; the latter can certify a contribution without independent review. The issue-reference share records numbered issues (e.g., "fixes #123"), and the PR-link share records pull-request references. The tracker-reference share combines these with external ticket keys, such as those from Jira. The merge or squash share records integration markers. The off-hours and weekend shares count commits authored between 20:00 and 05:59 UTC or on weekends in UTC. Joint tests do not reject zero pre-treatment coefficients for the reported measures.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

Second, development activity shifts toward regular working hours. Off-hours and weekend commit shares fall by 2.0 and 1.6 percentage points, respectively, both significant at the 5% level. The timing change is relevant to quality because contributions made at these times are more likely to be linked to later bug fixes. Within engineer and repository, off-hours and weekend indicators are associated with increases of 0.86 and 0.61 percentage points in the probability of attribution to a fix within 24 months, about 5% and 3% of the 17.7% base rate in the commit-level sample. Appendix M reports these regressions and the contribution of the timing shift to the overall defect decline. The movement toward regular hours provides further evidence of a more organized development process.

Third, the quality improvement extends to contributions made by engineers already present

before financing. For each identified bug fix, Git's line-attribution tool (`git blame`) finds the earlier commits that last modified the repaired lines. We classify these candidate defect-introducing commits by whether their authors were incumbents or newcomers.[11] In the firm–month attribution panel, 13.5% of commits are linked to a subsequent bug fix before the first round. The incumbent defect-attribution rate falls by 1.76 percentage points after financing ($p = 0.048$), roughly 13% of that base rate (Table 37). For newcomers, the differential is +0.54 percentage points and statistically insignificant ($p = 0.387$). This pattern is consistent with improvements in how existing developers work, and the retention tests in Appendix N find no detectable pattern of selective retention.

The contribution records make the professionalization channel concrete. Teams use more documented review, tracking, and integration procedures, organize more work within regular hours, and produce incumbent contributions that are less frequently linked to subsequent fixes. These changes fit a governance interpretation in which first institutional entry strengthens the procedures through which the existing team builds and maintains the product. The evidence thus connects the reduction in measured engineering defects to observable changes inside the development process.

4 Robustness

4.1 Matching on Pre-Treatment Growth

A concern with the baseline comparison is that VC-backed firms are already growing faster before financing, as Figure 8 shows for both headcount and code volume. Faster growth may accompany managerial capabilities or product demand that also improve engineering outcomes. We address this concern by matching treated firms to controls with similar pre-treatment headcount growth. The resulting comparison asks whether quality improves after first VC entry even relative to other fast-growing startups that followed comparable trajectories but remained without institutional financing throughout the event window.

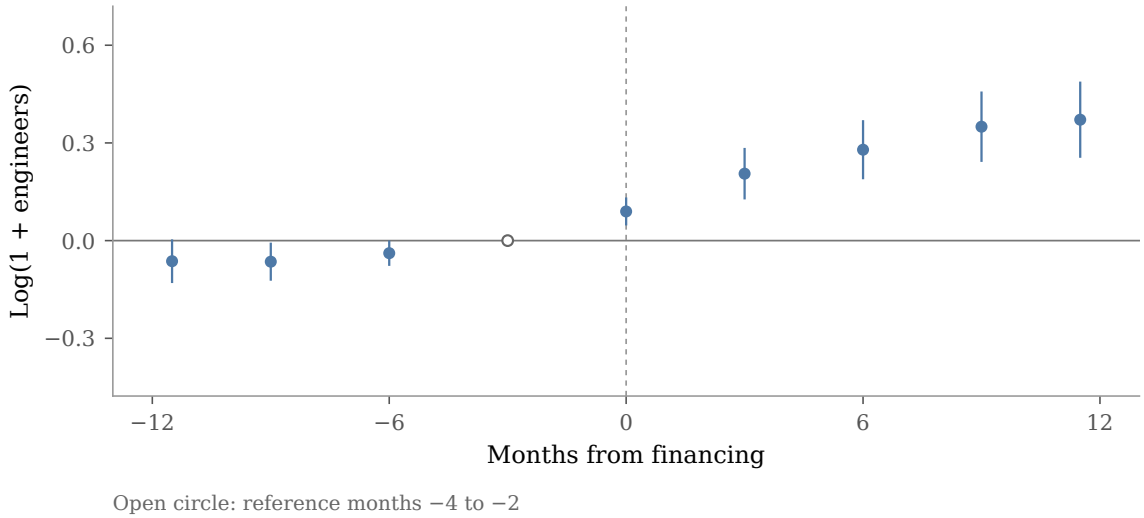
Within each cohort, we retain controls whose pre-treatment headcount growth is close to the average growth of the treated firms. We measure growth as the difference in log average headcount between relative months $[-4, -2]$ and $[-12, -11]$:

$$G_i^c = \ln\left(\frac{1}{3}\sum_{t=-4}^{-2} \text{Headcount}_{i,c,t}\right) - \ln\left(\frac{1}{2}\sum_{t=-12}^{-11} \text{Headcount}_{i,c,t}\right).$$

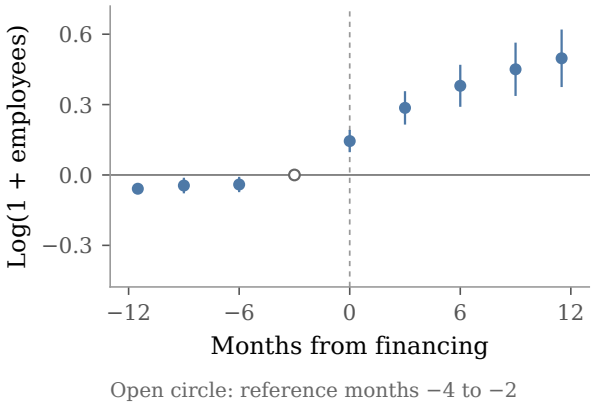
Appendix D details the matching procedure. Matching to the treated-group average yields controls that resemble the typical newly financed startup in the cohort, while preserving a broader comparison group than one-to-one matching.

[11] Attribution identifies earlier changes associated with subsequently repaired lines; its coverage depends on the discovery and documentation of bug fixes. It does not imply that every attributed change caused the later defect.

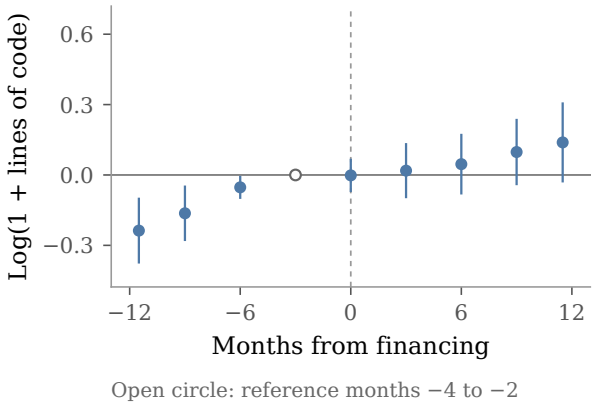
Figure 11 shows that matching substantially improves the comparability of growth trajectories. Before financing, software-engineering headcount, total headcount, and code volume show no systematic divergence between treated firms and their matched controls. After financing, headcount rises in treated firms relative to these fast-growing controls.



(a) Log(Software Engineers)



(b) Log(Total Employees)



(c) Log(Lines of Code)

Figure 11: Firm and Product Scaling (Matched Sample). Matched on pre-financing headcount growth within cohorts. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

Table 12 reports increases of 38 log points in total headcount and 29 log points in software-engineering headcount, both significant at the 1% level. Post-financing expansion remains when we compare treated firms with controls on similar prior growth trajectories. The quality comparison thus accounts for the growth characteristic most clearly associated with selection into VC financing.

Table 12: Difference-in-Differences Estimates for Firm Scaling (Matched Sample)

	Log(Total Headcount)			**Log(SWE Headcount)**		
	(1)	(2)	(3)	(4)	(5)	(6)
Treated × Post	0.377***	0.397***	0.433***	0.293***	0.325***	0.267***
	(0.043)	(0.060)	(0.059)	(0.044)	(0.061)	(0.054)
Post × High Violations (=1)		0.002			0.024	
		(0.019)			(0.019)	
Treated × Post × High Violations (=1)		−0.039			−0.058	
		(0.082)			(0.084)	
Post × Large Round			0.121***			0.068
			(0.037)			(0.046)
Treated × Post × Large Round			−0.224**			0.013
			(0.094)			(0.088)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	68,042	68,042	68,042	68,042	68,042	68,042
R^2	0.997	0.997	0.997	0.996	0.996	0.996

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Treated × Post indicator. The outcomes are the natural logarithms of one plus total headcount and software-engineering headcount. The sample is restricted to treated firms and control firms matched on pre-treatment headcount growth. See Appendix D for details on the matching procedure.
See Table 3 for the definitions of the High Violations and Large Round dummies.
Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

The engineering-quality estimates remain similar after matching (Table 13). The matching procedure primarily trims the control group by restricting attention to control firms with comparable pre-treatment growth dynamics; the matched stacked sample contains approximately 72,000 observations compared with 488,000 in the baseline—a reduction of roughly 85%. Despite this substantial reduction in sample size, the point estimates remain broadly stable. For critical violations, the matched coefficient (−2.32, significant at the 10% level) is 126% of the baseline estimate (−1.85), with a standard error that rises by 42% in the smaller sample. Total violations (−4.80) and maintainability issues (−4.41) are within 10% of their baseline magnitudes (−4.74 and −4.76), and both remain significant at the 5% level. Overall, the engineering-quality results are similar in magnitude when comparing treated firms to fast-growing controls.

Table 13: Difference-in-Differences Estimates for Engineering Quality (Matched Sample)

	Critical Violations			**Violations (All)**			**Maintainability Issues**		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Treated × Post	−2.321*	0.485	−2.764	−4.804**	−1.636	−4.720*	−4.410**	−1.412	−4.513*
	(1.223)	(0.833)	(1.705)	(2.373)	(1.206)	(2.443)	(2.107)	(1.115)	(2.295)
Post × High Violations (=1)		−1.798***			−4.577***			−4.160***	
		(0.447)			(1.670)			(1.468)	
Treated × Post × High Violations (=1)		−5.571*			−6.562			−6.191	
		(3.227)			(4.548)			(4.191)	
Post × Large Round			−0.424			−2.484*			−2.351*
			(0.506)			(1.329)			(1.209)
Treated × Post × Large Round			1.351			1.667			2.003
			(2.909)			(4.633)			(4.177)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	71,837	71,837	71,837	71,837	71,837	71,837	71,837	71,837	71,837
R^2	0.876	0.877	0.876	0.866	0.867	0.866	0.862	0.863	0.862

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Treated × Post indicator. The sample is restricted to treated firms and control firms matched on pre-treatment headcount growth. Critical Violations, Violations (All), and Maintainability Issues are measured per 1,000 lines of code. See Appendix D for details on the matching procedure and Appendix A, Table 19, for variable definitions.

See Table 3 for the definitions of the High Violations and Large Round dummies.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

The technical-debt estimates retain much of their baseline magnitude, although statistical significance falls in the smaller sample. Table 14 reports a decline of $1.29 per line, 82% of the baseline estimate, and a 4.70 percentage-point decline in the Technical Debt Ratio. Neither is statistically significant, with standard errors roughly 28% larger than in the full sample. For these outcomes, the loss of significance reflects both a smaller point estimate and the lower statistical power of the restricted comparison.

Table 14: Difference-in-Differences Estimates for Technical Debt (Matched Sample)

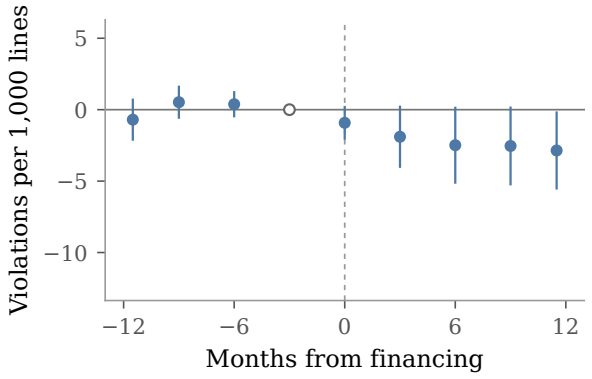
	Technical Debt per LoC			**Technical Debt Ratio**		
	(1)	(2)	(3)	(4)	(5)	(6)
Treated × Post	−1.291	−0.093	−0.931	−4.695	−0.338	−3.385
	(0.981)	(0.592)	(1.074)	(3.567)	(2.154)	(3.905)
Post × High Violations (=1)		−1.579**			−5.743**	
		(0.769)			(2.797)	
Treated × Post × High Violations (=1)		−2.466			−8.966	
		(2.118)			(7.702)	
Post × Large Round			−0.884			−3.214
			(0.537)			(1.952)
Treated × Post × Large Round			−0.177			−0.645
			(1.888)			(6.865)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	71,837	71,837	71,837	71,837	71,837	71,837
R^2	0.822	0.823	0.822	0.822	0.823	0.822

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$ on firms matched by pre-treatment headcount growth. Technical Debt per Line of Code is calibrated dollars per line; the Technical Debt Ratio is calibrated technical debt as a percentage of software assets. See Section 2.5 and Appendix D.

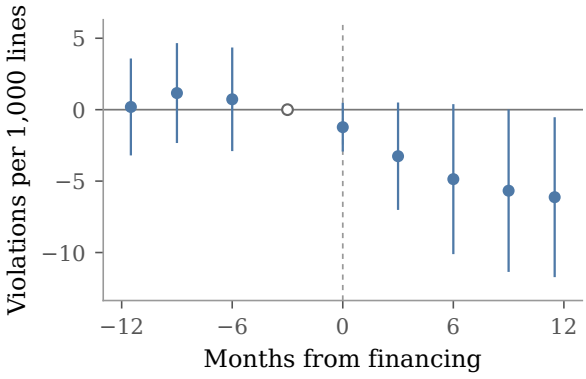
See Table 3 for the definitions of the High Violations and Large Round dummies.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

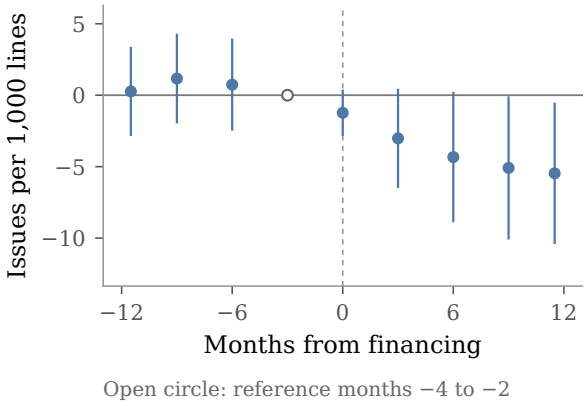
The matched event studies in Figures 12 and 13 reinforce the timing pattern in the baseline estimates. Pre-investment coefficients display no systematic divergence, followed by negative quality estimates after financing. The confidence intervals widen as the sample contracts, but the direction and timing of the engineering-quality response remain similar.



(a) Critical Violations per 1,000 LoC

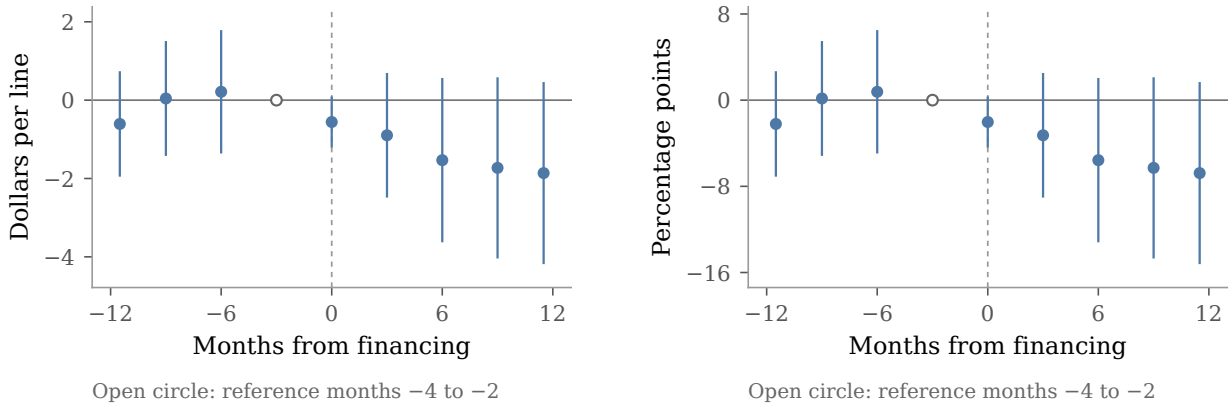


(b) Violations (All) per 1,000 LoC



(c) Maintainability Issues per 1,000 LoC

Figure 12: Engineering Quality (Matched Sample). Matched on pre-financing headcount growth within cohorts. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.



(a) Technical Debt per Line of Code

(b) Technical Debt Ratio

Figure 13: Technical Debt (Matched Sample). Matched on pre-financing headcount growth within cohorts. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

Overall, the matching analysis supports the robustness of the main quality findings. Critical violations remain significant at the 10% level, total violations and maintainability issues remain significant at the 5% level, and technical debt retains most of its estimated decline. The improvement therefore persists in a comparison that substantially narrows the observed differences in pre-financing growth, making those differences an unlikely explanation for the baseline quality pattern.

To assess whether these findings depend on the matching choices, we use a specification-curve analysis (Simonsohn, Simmons, and Nelson, 2020) over 72 configurations that vary the feature set (headcount only; headcount plus lines of code; headcount plus software engineering headcount; all three), the matching mode (centroid, individual $K = 1$, individual $K = 3$), and the caliper ($\delta \in \{0.5, 0.75, 1.0, 1.25, 1.5, 2.0\}$ units of Mahalanobis distance). Across all 72 specifications, every critical-violation, total-violation, and maintainability coefficient is negative; 85%, 74%, and 79%, respectively, reach the 10% significance level, and 26%, 61%, and 61% reach the 5% level. Technical-debt coefficients are negative in 68 of the 72 specifications. None reaches the 5% level, and six reach the 10% level. Appendix E reports the specification curves and full details.

4.2 Series B+ Placebo: Later Financing Rounds

A second concern is that the engineering improvements reflect only the relaxation of financial constraints: capital allows firms to hire engineers, purchase tools, and devote resources to remediation. This explanation has a testable implication. If an additional injection of resources produces recurring quality improvements, similar gains should appear around later financing rounds as well. We examine this prediction at the first Series B-or-later round, when firms receive further financing after institutional investors have already entered.

By contrast, the governance interpretation predicts a different timing pattern. First institutional entry can establish board oversight, operating routines, and professional standards that persist beyond the initial round (Hellmann and Puri, 2002; Kaplan and Strömberg, 2003). Once these practices are in place, a later capital injection need not reinstall them or generate another discrete quality gain. The contrast between financing stages therefore helps assess whether the observed improvement is specific to the initial organization of development under institutional backing.

We implement the placebo by re-centering the stacked difference-in-differences design on each firm's first completed Series B-or-later round recorded in PitchBook (Series B, C, D, and subsequent rounds). As in the main analysis, each cohort covers the 25 months from $e = -12$ through $e = +12$ relative to financing.

For each Series B+ cohort, controls are firms with no Seed or Series A round through the end of the event window. This restriction prevents first institutional entry among controls from affecting the later-round comparison.

We estimate the same fixed-effects specification as in the main analysis, replacing the treatment indicator with a Series B+ treatment indicator:

$$M_{i,c,t} = \beta^{B+} VC^{B+}_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}, \tag{10}$$

where B_i denotes firm i's cohort month of its first Series B+ round and $VC^{B+}_{i,c,t} \equiv \mathbf{1}\{B_i = c\} \cdot \mathbf{1}\{t \geq c\}$. As before, $\chi_{c,i}$ and $\delta_{c,t}$ are cohort $\times$ firm and cohort $\times$ month fixed effects, respectively. Outcomes $M_{i,c,t}$ include the same engineering-quality measures (e.g., critical violations, maintainability issues, technical debt, and related density-normalized metrics) as well as scaling outcomes (total headcount and software engineering headcount). Standard errors are clustered by cohort and firm, consistent with the baseline design.

Figure 14 compares the first-entry and later-round estimates on the same scale within each outcome. Headcount rises in both comparisons, whereas the later-round quality estimates are closer to zero. Figure 15 plots the later-round event studies for code volume in Panel (a), total headcount in Panel (b), and software-engineering headcount in Panel (c).

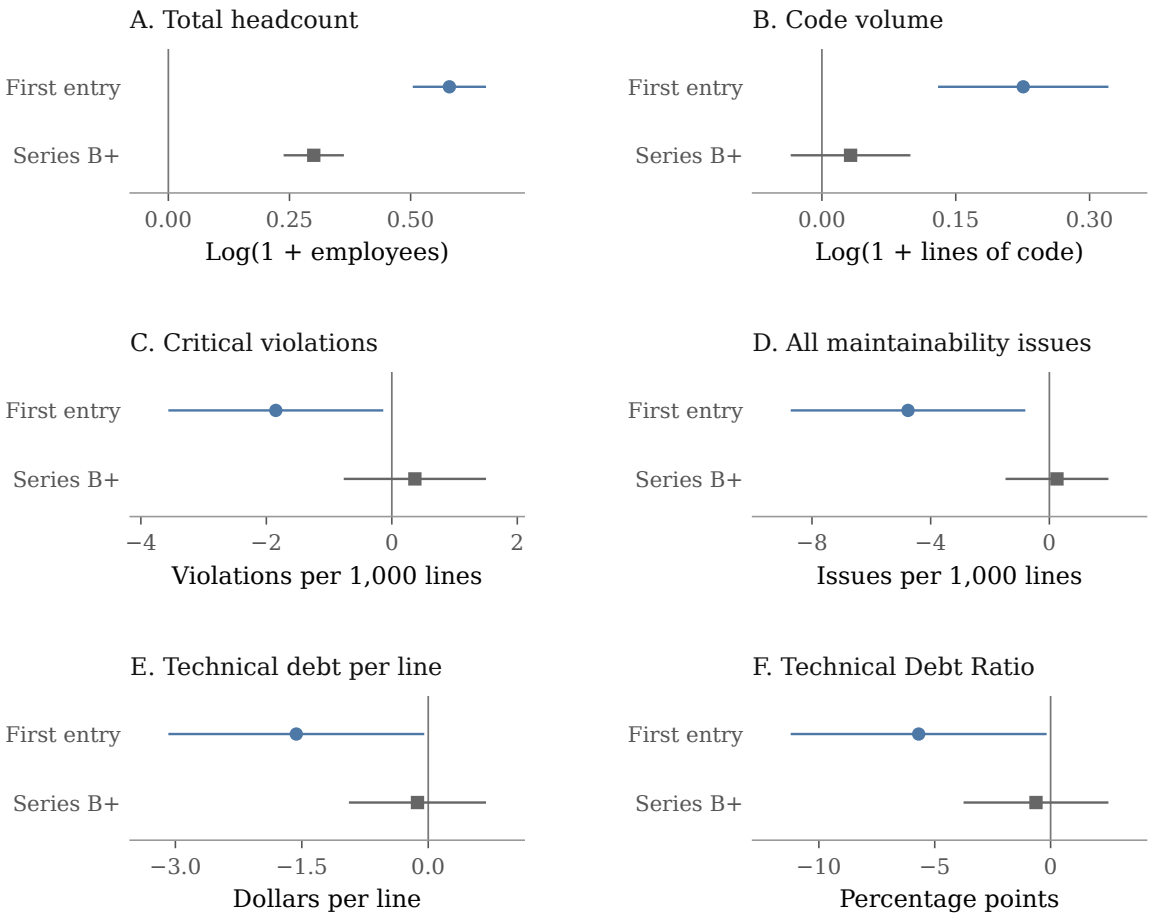
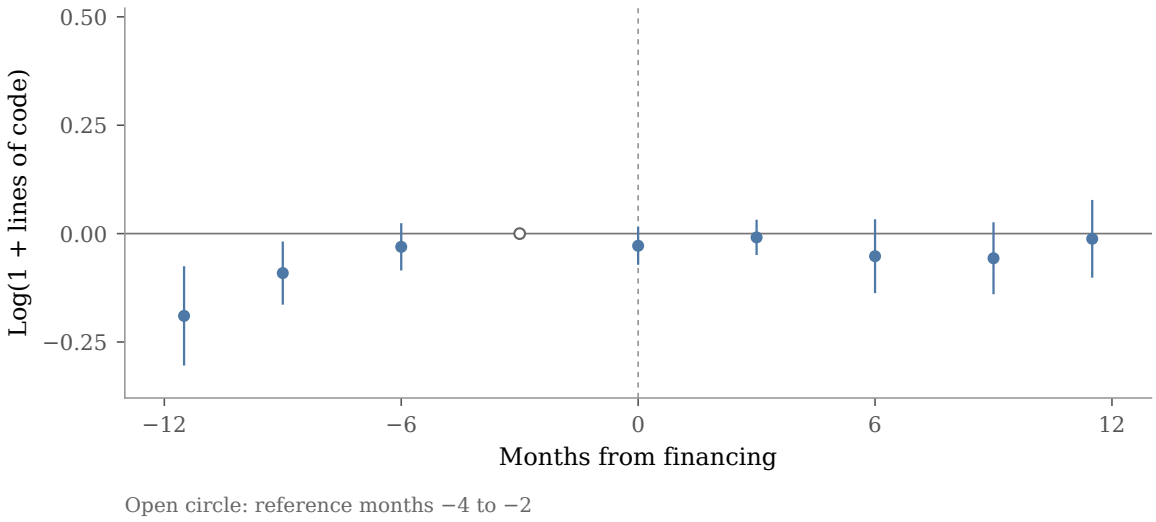
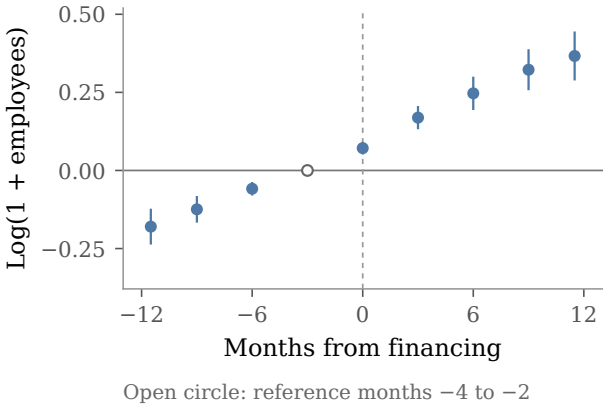


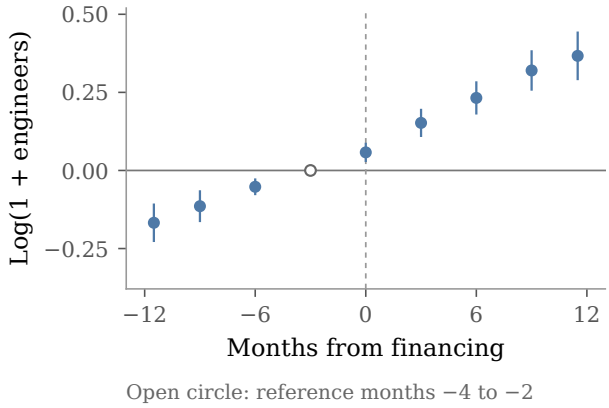
Figure 14: First Institutional Entry and Later Financing Rounds. Separate estimates within 12 months of first entry (circles) and the first Series B-or-later round (squares). Controls remain without first institutional financing through each window. Units appear below each panel. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.



(a) Log(Lines of Code)



(b) Log(Total Employees)



(c) Log(Software Engineers)

Figure 15: Firm and Product Scaling (Series B+ rounds). Controls remain without first institutional financing through the event window. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

Panels (b) and (c) show substantial headcount expansion after later financing, including a rise in the number of software engineers. Code volume in Panel (a), however, shows no statistically significant increase. This combination is informative for the resource explanation: firms add engineering capacity without a comparable expansion in the codebase, yet the quality measures below do not improve. The absence of a quality gain is therefore not confined to a setting in which rapid additions to code might offset improvements in existing software.

Despite this expansion, we find no corresponding improvement in engineering quality around later rounds. The estimates are small and statistically insignificant, often with the opposite sign to the

first-entry estimates, and the event studies show no clear break in quality. Table 15 summarizes the contrast; Appendix J reports the full estimates and plots.

Table 15: Baseline versus Series B+ Placebo Coefficients

Outcome	Baseline	Series B+ placebo	Source
Critical Violations	−1.849**	0.368	Table 28
Violations (All)	−4.744**	0.131	Table 28
Maintainability Issues (All)	−4.763**	0.256	Table 29
Maintainability Issues (High)	−1.840**	0.375	Table 29
Maintainability Issues (Low)	−0.727	−0.386	Table 29
Security Hotspots	−1.183	0.074	Table 30
Bugs	−0.007	−0.132	Table 31
Reliability Issues (High)	−0.006	−0.011	Table 31
Reliability Issues (Low)	−0.119	−0.148	Table 31
Technical Debt	−1.566**	−0.128	Table 32
Technical Debt Ratio	−5.693**	−0.631	Table 32

This table compares the main Treated × Post coefficient for each outcome in the baseline Seed/Series A specification with the corresponding coefficient from the Series B+ placebo specification. Baseline coefficients come from Tables 3, 4, 5, 6, and 7. Red text identifies the later-round placebo estimates. * p<0.1, ** p<0.05, *** p<0.01.

This asymmetry supports a role for technical discipline beyond capital availability alone. First VC entry accompanies both organizational expansion and a reduction in engineering defects; later rounds accompany substantial hiring without comparable quality improvements. The pattern is consistent with governance and professionalization standards being established when the first institutional investor enters and persisting through subsequent rounds. Because later-stage firms can face different constraints and the controls remain without a first round, we interpret the exercise as a comparison of financing-stage dynamics, with the mechanism inference supported by the process evidence in Section 3.8.

4.3 Lifecycle Stage and Commercialization

The first institutional round may also coincide with commercialization. As firms begin serving paying customers, they may adopt more demanding engineering practices. If the initial transition to revenue

generation explains our findings, quality gains should be concentrated among firms making that transition.

We test this prediction using commercial status at financing: 183 of the 274 treated firms (67%) were already generating revenue at first entry, and the estimated quality gains are not larger among the remaining pre-commercial firms. A complementary split by Seed versus Series A entry also fails to reject equality, including for critical violations ($p = 0.72$). Appendix L reports the round-type comparisons and their pre-trend diagnostics; we focus here on commercial status because it measures the transition of interest more directly.

Table 16 shows that several main outcomes improve among firms already generating revenue. Within this group, total violations decline by 5.92 per 1,000 lines, technical debt by $2.23 per line, and the Technical Debt Ratio by 8.11 percentage points, all significant at the 5% level. Critical violations decline by 1.90 per 1,000 lines among already-commercial firms and by 1.74 among pre-commercial firms; the first estimate is statistically insignificant and the second significant at the 10% level.

Table 16: Treatment Effects by Commercial Stage at Entry: Subsample Estimates

	Critical Violations	**Violations**	**Technical Debt**	**Technical Debt Ratio**
	(1)	(2)	(3)	(4)
Panel A: Treated firms already commercial at entry (183 firms)				
Treated × Post	−1.897	−5.920**	−2.231**	−8.114**
	(1.196)	(2.899)	(1.074)	(3.907)
# Obs	390,179	390,179	390,179	390,179
Panel B: Treated firms pre-commercial at entry (91 firms)				
Treated × Post	−1.743*	−2.400	−0.244	−0.886
	(0.949)	(1.454)	(0.540)	(1.963)
# Obs	297,099	297,099	297,099	297,099
Cohort × Firm FE	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes

Each panel estimates Equation (7) on treated firms with the stated commercial stage and the controls in their cohorts. Commercial stage is the PitchBook business status at the deal closest to treatment, matched within three months. Already commercial denotes Generating Revenue, Generating Revenue/Not Profitable, or Profitable. Cohorts left with only treated or only control firms are dropped. Critical Violations and Violations are counts per 1,000 lines of code; Technical Debt is calibrated dollars per line; the Technical Debt Ratio is calibrated technical debt as a percentage of software assets. Joint lead tests show no evidence of differential pre-treatment trends (smallest $p = 0.34$).

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

Table 17 directly tests whether the response differs by commercial stage. The pre-commercial interaction is statistically insignificant for critical and total violations. For both debt measures, however, it is positive at the 10% level, indicating a smaller decline among pre-commercial firms. Excluding

firms with more than five repositories yields similar estimates. The predicted concentration of gains among firms entering commercialization is therefore absent; for technical debt, the direction of the difference is the opposite.

Table 17: Commercial Stage at Entry: Triple-Difference Estimates

	Critical Violations	**Violations**	**Technical Debt**	**Technical Debt Ratio**
	(1)	(2)	(3)	(4)
Panel A: Main sample				
Treated × Post	−1.897	−5.919**	−2.230**	−8.110**
	(1.195)	(2.895)	(1.073)	(3.901)
Treated × Post × Pre-commercial	0.141	3.488	1.973*	7.176*
	(1.501)	(3.162)	(1.131)	(4.113)
Implied effect, pre-commercial firms	−1.755*	−2.431*	−0.257	−0.934
	(0.946)	(1.449)	(0.538)	(1.956)
# Obs	487,577	487,577	487,577	487,577
Panel B: At most five repositories per firm				
Treated × Post	−1.908	−6.012**	−2.275**	−8.274**
	(1.233)	(2.991)	(1.075)	(3.910)
Treated × Post × Pre-commercial	0.120	3.563	2.042*	7.425*
	(1.545)	(3.264)	(1.133)	(4.119)
# Obs	447,156	447,156	447,156	447,156
Cohort × Firm FE	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes

This table estimates $M_{i,c,t} = \beta_1 VC_{i,c,t} + \beta_3 \left(VC_{i,c,t} \times \text{Pre-commercial}_i\right) + \chi_{c,i} + \delta_{c,t} + \varepsilon_{i,c,t}$ on the full estimation sample, retaining every eligible control firm. Firms *already commercial* at entry are the omitted category: β_1 (Treated × Post) is the treatment effect for firms already generating revenue when the first institutional round closes, and β_3 is the incremental effect for pre-commercial firms. Pre-commercial equals one for treated firms not yet generating revenue at the financing deal closest to the treatment date (PitchBook deal business status, matched within three months) and zero for treated commercial firms; it is identically zero for control firms, whose commercial stage at the cohort baseline is unobservable, so the specification estimates treatment-effect heterogeneity across treated firms on the common control pool rather than a fully saturated triple difference. The level and Post interactions of the indicator are absorbed by the fixed effects or collinear with the triple term and are not separately identified. The implied effect for pre-commercial firms is $\beta_1 + \beta_3$; its standard error comes from the equivalent reparameterization with pre-commercial firms as the base group. Joint tests show no evidence of differential pre-treatment leads by commercial stage for any of the four outcomes (smallest $p = 0.79$). Panel B drops firms with more than five scanned repositories before aggregation. Critical and total violations are counts per 1,000 non-comment lines of code; technical debt is calibrated dollars per line, and the Technical Debt Ratio is calibrated debt as a percentage of software replacement cost.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

Commercialization can develop gradually even after a firm begins generating revenue. We therefore complement the stage comparison with direct measures of how the software asset changes around financing. Table 18 uses developers' descriptions of code changes (commit messages) and the paths of the files they alter to track references to billing, payments, subscriptions, trials, and upgrades. We also measure changes to commercial, payment-library, enterprise, and growth-related code paths,

while Panel C records the scale and breadth of development. These outcomes use the single-repository sample described in the table notes.

Table 18: Commercialization Measures in the Software Asset

Measurement	Dependent Variable	Treated × Post	p-value
Panel A: Commit-message language			
	Any commercialization term	0.0070	0.348
		(0.0074)	
	Billing / payments / metering	0.0001	0.953
		(0.0011)	
	Subscription / trial / upgrade	0.0065	0.374
		(0.0073)	
Panel B: Changed file paths			
	Commercial code paths	−0.0003	0.332
		(0.0003)	
	Payment-library paths	−0.0004	0.362
		(0.0005)	
	Enterprise / SLA paths	0.0014	0.335
		(0.0014)	
	Growth / expansion paths	0.0012	0.610
		(0.0024)	
Panel C: Development activity			
	Log file changes per month	0.3374***	<0.001
		(0.0660)	
	Top-level directories touched	0.8844**	0.022
		(0.3778)	
	Frontend share of changed files	0.0177**	0.043
		(0.0086)	

This table estimates the baseline specification of Equation (7) on commit-derived outcomes for the single-repository panel. Panel A uses 4.8 million commit messages (177,676 stacked observations); each outcome is the fraction of a firm's commits in a month carrying the stated marker. Panels B and C use file-change logs from 441 cloned repositories. Panel B measures the fraction of changed files whose paths match each category. Panel C measures log(1 + file changes), the number of top-level directories touched, and the fraction of changed files in the user-facing interface. Joint tests do not reject zero pre-treatment coefficients for the reported measures. These outcomes derive from commit histories and contain no SonarQube input.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

The monetization markers show no statistically significant shift after financing, while development activity expands substantially. Monthly file changes rise by 34 log points, developers touch 0.88 additional top-level directories, and the share of changed files in the user-facing interface rises by 1.77

percentage points, each significant at the 5% level or better. The increase in development therefore extends across the software asset without a detectable shift toward the measured monetization features.

Employment records show a similar expansion in engineering capacity. The sales-related share of headcount rises by 1.50 percentage points, compared with 6.65 percentage points for software engineering. Firms build commercial capacity, but the larger compositional shift is toward engineering. Appendix K reports the employment decomposition and additional development diagnostics.

The stage, codebase, and employment tests do not support a narrow lifecycle explanation in which quality improves because firms make the initial transition to selling a product. The gains are present among firms already generating revenue and accompany broad development activity, with a larger employment shift toward engineering than toward sales. These findings complement the process evidence by showing that the engineering improvement extends beyond the firms and activities most directly associated with the onset of commercialization.

4.4 Additional Robustness Tests

We next examine whether the quality improvement depends on the normalization of engineering outcomes, the months surrounding the financing event, or particular firms and cohorts in the stacked sample. These tests address distinct features of the design, while supplementary outcomes show which dimensions of engineering practice improve and which do not.

Density versus levels. A decline in violations per line could arise mechanically if code volume expands faster than the stock of defects. We therefore examine the result using both a low-growth subsample and a specification that models counts directly. Among firms in the bottom quartile of pre-financing code growth, the critical-density estimate strengthens to -2.85 per 1,000 lines, while a Poisson model conditioning on code volume yields 8.5% fewer critical violations. The improvement thus survives two different treatments of scale. Appendices G and H present these tests and connect the density results to the evolution of defect stocks.

Donut event study. If preparation for due diligence or the timing of the recorded financing month accounts for the result, excluding treatment-adjacent observations should weaken it. Instead, removing observations one and two months on either side of financing increases the magnitude of the critical-violation estimate to -1.970 (SE = 0.943) and -2.065 (SE = 0.986), respectively. Technical-debt and total-violation estimates also remain negative and significant at the 5% level. The persistence and modest strengthening of the estimates indicate that the improvement extends beyond the months immediately surrounding the deal.

Window sensitivity. The findings also persist across alternative event windows. With ±6-, ±9-, ±12-, ±18-, and ±24-month windows, critical-violation estimates range from −2.30 to −1.71. They are significant at the 5% level through ±18 months and at the 10% level at ±24 months ($p = 0.05004$). Total violations are significant at the 5% level in every window. Technical debt is significant at the 5% level except at ±6 months ($p = 0.061$). The main results therefore do not depend on the choice of a 12-month window.

Leave-one-cohort-out and permutation tests. No single financing cohort drives the results. Dropping each of the 99 cohorts in turn leaves every coefficient negative: critical violations range from −2.00 to −1.25, technical debt from −1.67 to −1.06, and total violations from −5.00 to −3.15. Furthermore, in 500 within-cohort treatment permutations, no critical-violation draw is as large in absolute value as the observed estimate. The reported permutation p-values are below 0.002 for critical violations, 0.002 for total violations, and 0.028 for technical debt, compared with two-way-clustered asymptotic p-values of 0.035, 0.021, and 0.043.[12]

Control reuse in the stacked design. The 412 control firms appear in an average of 49.8 cohorts each, with a maximum of 99. We cluster by firm to account for dependence across repeated appearances and assess whether heavily reused controls drive the estimates. Weighting observations by the inverse of each control firm's appearance frequency yields coefficients of −1.976 for critical violations (SE = 0.892), −1.830 for technical debt (SE = 0.917), and −5.295 for total violations (SE = 2.523), all significant at the 5% level. Alternatively, capping reuse at 50 cohorts retains 227 control firms and roughly a quarter of observations; critical violations are −1.746 ($p = 0.07$) and total violations −5.291 ($p = 0.06$). Stricter caps retain at most 7% of observations and provide limited information. The estimates remain similar under the more informative restrictions, reducing concern that a few frequently reused controls account for the findings.

Language heterogeneity. The five most common dominant languages among treated firms are Go (66 firms), JavaScript (54), C/C++ (46), Java (45), and Python (45). To assess whether any one language accounts for the result, we exclude each group in turn. Every critical-violation coefficient remains negative, ranging from −0.56 without JavaScript-dominant firms to −2.44 without Go-dominant firms. Four of the five estimates are significant at the 5% level; excluding C/C++ yields −1.65 (SE = 1.01), which does not reach the 10% level. Total-violation and technical-debt coefficients also remain negative in every exclusion, although total violations lose 5% significance without C/C++ or JavaScript, and technical debt loses it without C/C++, JavaScript, or Python. The direction of the result is therefore stable across language exclusions, with its magnitude and statistical significance varying across the remaining samples.

[12] The permutation reference distribution exchanges treatment labels within financing cohorts.

Additional SonarQube outcomes. A supplementary screen of 42 previously unused outcomes clarifies the scope of the result. The *code smells* taxonomy yields −4.803 issues per 1,000 lines (SE = 1.990, $p < 0.05$), closely matching the maintainability estimate. By contrast, log duplicated lines rise by 0.358 (SE = 0.099, multiplicity-adjusted $q = 0.001$), and duplicated blocks rise by 0.942 per 1,000 lines (SE = 0.448), although the latter is not significant after adjustment ($q = 0.114$). Security measures remain statistically insignificant. Improved rule compliance and maintainability thus coexist with increased duplication as firms scale.

Process adoption and repository allocation. We also examine whether the process changes extend to sustained testing and continuous integration—procedures that automatically build and test code updates. The monthly probability of adopting these practices rises by 0.224 percentage points relative to a 0.47% base ($p = 0.046$), and a fixed-effects logit gives an odds ratio of 1.38 (95% CI [1.02, 1.88]). This evidence is suggestive because adjustment across the four adoption markers removes conventional significance. Separately, neither the commit share of the flagship repository—the firm's largest repository before financing—nor within-firm commit concentration changes detectably (+0.031, $p = 0.321$, and −0.006, $p = 0.633$, respectively). The allocation results provide no evidence that the improvement simply reflects a shift of development toward the flagship repository. Appendix N reports the sample restrictions and full results.

5 Conclusion

This paper examines whether venture capital investment is linked with changes in the technical practices through which startups build their products. Using longitudinal codebase data on 569 open-core technology firm's products, we find notbale improvements across a wide range of code quality metrics. These improvements occur as firms expand their codebases and engineering teams. The decline in defect density we report persists when we compare firms with similar prior growth, restrict attention to firms with low pre-financing code growth, and control explicitly for code volume. Quality improvements also extend to firms already generating revenue.

Our analysis of individual code contributions provides evidence on the changes that may drive this shift in quality after VC entry. Teams document more code review and issue tracking and conduct a larger share of their work during regular hours. Furthermore, a smaller share of contributions by incumbent engineers is linked to later bug fixes. Our results offer little support for the view that recruiting senior technical leaders is the principal explanation for the quality gains. Furthermore, we document that later financing rounds also lead to substantial scaling but no statistically significant improvement in engineering quality. Our comparisons across investor types and syndicate sizes suggest a role for monitoring. Together, these findings suggest that VC professionalization extends to the technical practices of existing teams. Finally, our novel dollar-denominated measure of technical

debt also makes software maintenance liabilities tractable for research on software assets and firm value.

We acknowledge several limitations. VC financing is not randomly assigned, and investors may anticipate improvements that would have occurred without their involvement. Although the evidence points to changes in development practices, our comparisons do not identify the exact governance actions responsible for them. Finally, public scrutiny may make open-core firms respond differently to VC entry than startups whose software is proprietary. Direct evidence on investor interventions would help explain how investors influence engineering practices and whether these changes persist.

References

Amornsiripanitch, N., P. A. Gompers, and Y. Xuan. 2019. More than money: Venture capitalists on boards. *The Journal of Law, Economics, and Organization* 35:513–43.

Banker, R., Y. Liang, and N. Ramasubbu. 2021. Technical debt and firm performance. *Management Science* 67:3174–94.

Bernstein, S., X. Giroud, and R. R. Townsend. 2016. The impact of venture capital monitoring. *The Journal of Finance* 71:1591–622.

Bernstein, S., and A. Sheen. 2016. The operational consequences of private equity buyouts: Evidence from the restaurant industry. *The Review of Financial Studies* 29:2387–418.

Callaway, B., and P. H. Sant'Anna. 2021. Difference-in-differences with multiple time periods. *Journal of Econometrics* 225:200–30. Themed Issue: Treatment Effect 1.

Casamatta, C. 2003. Financing and advising: Optimal financial contracts with venture capitalists. *The Journal of Finance* 58:2059–85.

Casamatta, C., and C. Haritchabalet. 2007. Experience, screening and syndication in venture capital investments. *Journal of Financial Intermediation* 16:368–98.

Chemmanur, T. J., and Z. Chen. 2014. Venture capitalists versus angels: The dynamics of private firm financing contracts. *Review of Corporate Finance Studies* 3:39–86.

Eisfeldt, A. L., and D. Papanikolaou. 2013. Organization capital and the cross-section of expected returns. *The Journal of Finance* 68:1365–406.

Fershtman, C., and N. Gandal. 2011. Direct and indirect knowledge spillovers: the "social network" of open-source projects. *The RAND Journal of Economics* 42:70–91.

Gompers, P. A. 1995. Optimal investment, monitoring, and the staging of venture capital. *The Journal of Finance* 50:1461–89.

Gompers, P. A., W. Gornall, S. N. Kaplan, and I. A. Strebulaev. 2020. How do venture capitalists make decisions? *Journal of Financial Economics* 135:169–90.

Gorman, M., and W. A. Sahlman. 1989. What do venture capitalists do? *Journal of Business Venturing* 4:231–48.

Gormley, T. A., and D. A. Matsa. 2011. Growing out of trouble? corporate responses to liability risk. *The Review of Financial Studies* 24:2781–821.

Hellmann, T., and M. Puri. 2002. Venture capital and the professionalization of start-up firms: Empirical evidence. *The Journal of Finance* 57:169–97.

Hsu, D. H. 2004. What do entrepreneurs pay for venture capital affiliation? *The Journal of Finance* 59:1805–44.

Ibrahim, D. M. 2008. The (not so) puzzling behavior of angel investors. *Vanderbilt Law Review* 61:1405–52.

Kang, J.-K., Y. Li, and S. Oh. 2022. Venture capital coordination in syndicates, corporate monitoring, and firm performance. *Journal of Financial Intermediation* 50:100948–.

Kaplan, S. N., and P. Strömberg. 2003. Financial contracting theory meets the real world: An empirical analysis of venture capital contracts. *The Review of Economic Studies* 70:281–315.

Krasner, H. 2022. Cost of poor software quality in the U.S. : A 2022 report. Working Paper, Consortium for Information & Software Quality (CISQ). Accessed: March 25, 2025.

Lerner, J. 1994. The syndication of venture capital investments. *Financial Management* 23:16–27.

Letouzey, J.-L. 2012. The SQALE method for evaluating technical debt. In *2012 Third International Workshop on Managing Technical Debt (MTD)*, 31–6. Zurich, Switzerland. doi:10.1109/MTD.2012.6225997.

McKinsey, and Co. 2020. Tech debt: Reclaiming tech equity. *McKinsey & Company* Accessed: March 25, 2025.

———. 2023. Breaking technical debt's vicious cycle to modernize your business. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/breaking-technical-debts-vicious-cycle-to-modernize-your-business. Accessed: March 25, 2025.

Nagle, F. 2019. Open source software and firm productivity. *Management Science* 65:1191–215.

Peters, R. H., and L. A. Taylor. 2017. Intangible capital and the investment-q relation. *Journal of Financial Economics* 123:251–72.

Sculley, D., G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young. 2014. Machine learning: The high interest credit card of technical debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*.

Sculley, D., G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. 2015. Hidden technical debt in machine learning systems. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, 2503–2511. Cambridge, MA, USA: MIT Press.

Simonsohn, U., J. P. Simmons, and L. D. Nelson. 2020. Specification curve analysis. *Nature Human Behaviour* 4:1208–14.

Sun, L., and S. Abraham. 2021. Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. *Journal of Econometrics* 225:175–99. Themed Issue: Treatment Effect 1.

Appendix

A Variable Definitions

Table 19: Data and Variable Definitions

Variable	Description
A. Dimension of the data	
Firms	Open-core technology companies that commercialize open-source repositories and for which we observe longitudinal code histories, funding events, and employment data. The universe comprises 649 unique firms identified through OSSrank, an analytics platform for the open-source ecosystem; 621 enter the firm–month panel, 569 of them with covered codebase months.
Repositories	Core product repositories associated with each firm. The firm–month panel draws on 703 repositories hosted on GitHub or GitLab platforms over the period 2010–2024. A repository functions as the central storage location for the code underlying a firm's core product.
Month	Monthly observations covering the period 2010–2024.
Unit of observation	The source panel records one observation per firm–month. The estimation panel is a firm–cohort–month because a firm can appear in more than one treatment stack.
Firm–month aggregation	Extensive SonarQube metrics (lines of code, remediation effort, and issue counts by severity) are summed across a firm's repositories within a month. Ratio measures, including the Technical Debt Ratio, are recomputed from the summed components rather than averaged across repositories.
B. Dependent variables (Engineering quality metrics)	

Continued on next page

Table 19 (continued)

Variable	Description
Critical Violations	Number of SonarQube rule violations at the critical severity level per 1,000 non-comment lines of code in month t. These severe issues can threaten system stability, security, or future maintainability. SonarQube also has a small Blocker category, which is not the focus of the analysis.
Violations (All)	Number of rule violations across all severity levels per 1,000 non-comment lines of code. SonarQube flags code patterns associated with reliability, security, and maintainability concerns.
Maintainability Issues (All)	Number of maintainability-related rule violations across all severities per 1,000 non-comment lines of code. These rules flag code patterns that may make software harder to understand, modify, or extend.
Maintainability Issues (High)	Number of maintainability issues classified as high severity by SonarQube per 1,000 non-comment lines of code.
Maintainability Issues (Low)	Number of maintainability issues classified as low severity by SonarQube per 1,000 non-comment lines of code.
SonarQube Open Issues	Omitted from analysis. In retrospective scanning, every rule violation is simultaneously an open issue; the variable is therefore identical to Total Violations and is excluded to avoid double-counting.
kLoC	One thousand non-comment lines of source code. Issue densities use this unit; technical debt per LoC uses one line.
Lines of code (LoC)	Non-comment lines of source code summed across the firm's covered repositories in month t.
SonarQube Security Hotspots	Number of security-sensitive code segments flagged for manual review per 1,000 non-comment lines of code. A hotspot requires human assessment and is not a confirmed vulnerability.
Bugs	Number of code patterns classified by SonarQube as likely to cause incorrect or unexpected behavior, per 1,000 lines of code.

Continued on next page

Table 19 (continued)

Variable	Description
Reliability Issues (High)	Number of reliability issues classified as high severity by SonarQube per 1,000 non-comment lines of code. The classification indicates a risk of incorrect behavior, rather than an observed application failure.
Reliability Issues (Low)	Number of reliability issues classified as low severity by SonarQube per 1,000 non-comment lines of code.
Technical Debt per Line of Code	Estimated dollar value of remediation effort per line of code. Let w denote the wage per hour ($55/hour) and let the SQALE Index be SonarQube's remediation effort for maintainability issues (in hours). Let ℓ denote the development time per line (in hours) and set $\ell = 0.5$ (30 minutes), SonarQube's default. We translate lines of code into a re-development effort proxy $\ell \times \text{LoC}$ hours and translate the raw SQALE Index into calibrated remediation effort using κ. We choose κ so that median calibrated debt equals a target share φ of median replacement-cost software assets (we set $\varphi = 0.20$): $$\kappa = \varphi \times \frac{\text{Median}(\ell \times \text{LoC})}{\text{Median}(\text{SQALE Index})}.$$ We then define $$\text{Technical Debt}_{i,t} = \kappa \times \text{SQALE Index}_{i,t} \times w,$$ and normalize by lines of code: $$\text{Technical Debt per LoC}_{i,t} = \frac{\text{Technical Debt}_{i,t}}{\text{LoC}_{i,t}}.$$

Continued on next page

Table 19 (continued)

Variable	Description
Technical Debt Ratio	Calibrated Technical Debt divided by Software Assets, multiplied by 100, where Software Assets is the replacement cost of the codebase ($55 per hour times 30 minutes per line times lines of code). Equivalently, 100κ times raw remediation hours divided by estimated redevelopment hours; the wage cancels. The ratio is a percentage of replacement cost, and changes are reported in percentage points.
Technical Debt (level)	Calibrated dollar value of the remediation burden in month t, defined as $\kappa \times \text{SQALE Index}_{i,t} \times w$.
κ (Technical-debt calibration)	Scaling constant that maps SonarQube's raw SQALE Index into dollar-denominated technical debt. It is chosen so that median calibrated debt equals 20% of median replacement-cost software assets ($\varphi = 0.20$). The main-sample calibration is $\kappa = 68.60$ and is held fixed across specifications using that sample.
C. Main explanatory variables	
$VC_{i,c,t}$ (Treated × Post)	Indicator equal to 1 for a treated firm in cohort c beginning in the month of its first completed Seed round, or first completed Series A round if no Seed is recorded. Rumored, failed or cancelled, announced or in-progress, upcoming, and postponed deals are excluded when assigning the event.
Treated	Indicator equal to 1 for a firm whose first qualifying financing defines cohort c, and 0 for that cohort's controls. A firm can serve as a control in an earlier cohort before entering its own treated cohort.
Post	Dummy equal to 1 for months at or after the first VC deal for treated firms (and the corresponding aligned event-time months for controls in each cohort), and 0 otherwise.

Continued on next page

Table 19 (continued)

Variable	Description
High Violations	Dummy equal to 1 if the firm's pre-treatment critical violation density (at $t = -1$) exceeds the median across baseline firm–cohort observations of firms that appear as treated somewhere in the sample, and 0 otherwise. A firm can contribute baseline observations in more than one cohort.
Large Codebase	Indicator equal to one if log code volume at event month -1 exceeds the median across baseline firm–cohort observations of firms treated somewhere in the sample. The comparison uses the same baseline convention as High Violations.
Statically Typed Language	Indicator for a dominant language recorded as Go, Java, Rust, Scala, C#, C, Kotlin, or C++ in the analysis classification. The included source codes are `go`, `java`, `rust`, `scala`, `cs`, `c`, `kotlin`, and `cpp`; other recorded codes take value zero.
Large Round	Dummy equal to 1 if the VC deal size (first Seed or Series A round amount, in USD) exceeds the treated-firm median, and 0 otherwise. Used to compare quality changes across financing amounts.
Angel First Round	Dummy equal to 1 if the firm's first qualifying capital round among Angel (individual), Seed, and Series A rounds is an Angel (individual) round, and 0 otherwise. Used to compare quality changes at first Seed or Series A entry according to whether the firm previously received angel financing.
Syndicate (p75 or p50)	Dummy equal to 1 if the first Seed or Series A treatment round has more investors than a specified treated-firm investor-count threshold. The main monitoring specification uses the 75th percentile threshold (more than 8 investors), with the median threshold (more than 4 investors) reported as robustness.

Continued on next page

Table 19 (continued)

Variable	Description
Technical Founder	Dummy equal to 1 if any member of the founding team (identified via LinkedIn job titles containing "founder" or "co-founder" at the time of first VC financing) simultaneously held a technical leadership role (CTO, VP of Engineering, Head of Engineering, or Director of Engineering) at the same firm. Captures whether the founding team includes individuals who combine entrepreneurial and engineering expertise.
Serial Entrepreneur	Dummy equal to 1 if any founder of the firm held a prior CEO, Chief, Founder, or President role at a different company before the first VC deal date (excluding assistant and vice president roles). Captures prior startup experience within the founding team.
Series B+ placebo treatment	In the Series B+ placebo sample of mature firms, dummy equal to 1 in months at or after a Series B or later funding round (conditional on observing earlier Seed/Series A history), and 0 otherwise. Treated firms already have institutional backing; controls remain without a first Seed or Series A round through the end of the event window.
Already commercial	Dummy equal to 1 if the firm was already generating revenue when venture capital arrived, based on the PitchBook business status recorded at the financing deal that defines treatment (matched within ±3 months of the treatment date) and taking the value *Generating Revenue*, *Generating Revenue/Not Profitable*, or *Profitable*; 0 otherwise. Defined for treated firms.
Pre-commercial	Complement of *Already commercial* for treated firms: dummy equal to 1 if the firm was not yet generating revenue at the treatment deal, and 0 otherwise. Used to test whether the effect of VC entry differs by the firm's commercial stage at entry.
D. Other covariates and constructed variables	
Total firm headcount	Total number of employees associated with the firm based on LinkedIn data in month t.

Continued on next page

Table 19 (continued)

Variable	Description
Software engineer headcount	Number of employees with software-engineering roles (based on LinkedIn job titles) in month t.
New hires (24-month)	Total number of distinct new employees who started at the firm within 24 months of the first VC deal date and who were not active at the firm before funding. Identified from LinkedIn position records.
Senior engineering hires	Number of new hires (within 24 months post-deal) whose LinkedIn job titles indicate senior engineering roles: senior engineer/developer, staff engineer, principal engineer, architect, engineering director/VP, engineering manager, tech lead, or CTO.
Junior hires	Number of new hires (within 24 months post-deal) whose LinkedIn job titles indicate junior-level roles: junior, intern, entry-level, associate, graduate, or trainee positions.
Senior hire ratio	Ratio of senior engineering hires to total new hires in the 24-month post-deal window. Null for firms with zero new hires in the window.
High Senior Ratio	Indicator equal to one if the ratio of senior engineering hires to all new hires over the 24-month post-financing window exceeds the treated-firm median. Missing hiring measures are excluded from the interaction regressions.
Hired technical leader	Dummy equal to 1 if the firm hired a new CTO, VP of Engineering, or Head of Engineering within 24 months of the first VC deal.
New SWE hires	Number of new hires (within 24 months post-deal) with software engineering job titles: software engineer/developer, backend, frontend, full-stack, mobile developer, development-and-operations engineer, site reliability engineer, platform engineer, data engineer, or machine-learning engineer.

Continued on next page

Table 19 (continued)

Variable	Description
Technical-leader hiring outcome	Indicator equal to one if a firm hires a CTO, VP of Engineering, Head or Director of Engineering, or VP of Technology during the pre-financing period $[-12, -2]$ or post-financing period $[-1, 12]$. Unlike the 24-month moderator above, this outcome is defined separately in each period.
Nontechnical-leader hiring outcome	Analogous period-specific indicator for hiring a nontechnical executive (chief executive, operating, financial, marketing, product, revenue, or commercial officer, or a head, vice president, or director of a non-engineering function).
SWE Share	Software-engineering individual contributors as a share of distinct new hires in a firm–cohort–quarter, based on LinkedIn role classifications. Missing when no hires are recorded. SWE denotes software engineering.
SWE Seniority	Mean LinkedIn seniority, on a scale from 1 to 7, among new software-engineering individual contributors in a firm–cohort–quarter. Missing when no classified engineering hires are recorded.
$X_{i,c,t}$	Additional regressors in the heterogeneity specifications, including Post and Treated $\times$ Post interactions with baseline quality, financing size, and other stated moderators. The baseline regression omits $X_{i,c,t}$.
Log LoC	Natural logarithm of one plus non-comment lines of code summed across covered repositories.
Log total headcount	Natural logarithm of one plus total firm headcount.
Log software engineer headcount	Natural logarithm of one plus software-engineer headcount.

Continued on next page

Table 19 (continued)

Variable	Description
Pre-treatment headcount growth	Difference in log average headcount between months $t \in [-4, -2]$ and $t \in [-12, -11]$ relative to the VC event month for firm i in cohort c. Used in Mahalanobis-distance matching to select controls with similar observed pre-treatment growth: $$G_i^c = \ln\left(\frac{1}{3}\sum_{t=-4}^{-2} \text{Headcount}_{i,c,t}\right) - \ln\left(\frac{1}{2}\sum_{t=-12}^{-11} \text{Headcount}_{i,c,t}\right).$$
Pre-treatment SWE headcount growth	Analogous pre-treatment growth measure constructed using software engineer headcount instead of total headcount over the same pre-treatment windows.
Pre-treatment software growth	Analogous pre-treatment growth measure constructed using lines of code (LoC) over the same pre-treatment windows.
Software Assets	Estimated replacement cost of the codebase in month t. Software Assets represent a coarse approximation of the capital required to re-develop the codebase from scratch, defined using a replacement cost method: $$\text{Software Assets}_{i,t} = w\ell\,\text{LoC}_{i,t},$$ where $\ell = 0.5$ hours per line and $w = \$55$ per hour. This is a replacement-cost proxy, not observed expenditure.
Event-time indicators	Set of dummy variables for each quarter or month $t \in [-12, +12]$ relative to the VC event date. Used to estimate dynamic difference-in-differences (event-study) coefficients and assess pre-treatment outcome paths.
Cohort × Firm FE ($\chi_{c,i}$)	Firm-by-cohort fixed effects that absorb time-invariant firm characteristics within each stacked cohort.

Continued on next page

Table 19 (continued)

Variable	Description
Month × Cohort FE ($\delta_{c,t}$)	Month-by-cohort fixed effects that control for common time-varying shocks affecting all firms within a cohort at month t.
Cohort-integrity rule	Sample rule applied after every filter: any stacked cohort left with only treated firms, or only control firms, is dropped before estimation because it carries no identifying variation.

Note: Engineering-quality metrics are measured using SonarQube Community Edition static analysis, which checks code against more than 9,000 language-specific rules. Technical Debt uses the SQALE (Software Quality Assessment based on Lifecycle Expectations) methodology. The universe comprises 649 open-core technology firms identified through OSSrank. The 2010–2024 firm–month panel links 703 repositories to 621 firms; 569 firms contribute covered codebase months from our SonarQube scans. The stacked difference-in-differences sample contains 484 unique firms and 99 event cohorts. Among these firms, 274 appear as treated and 412 as controls; the counts overlap because a not-yet-treated firm can serve as a control before its own financing event. Outcomes are observed over $t \in [-12, +12]$ around the first qualifying financing.

B Language-Specific Analysis Configurations

To reconstruct engineering quality over time, we analyze historical source code using SonarQube Community Edition and its language analyzers, supplemented by the community *sonar-cxx* plugin for C and C++. Historical repositories often lack working build environments or compiled dependencies. We use the following language-specific settings to reduce reliance on those artifacts and accommodate large codebases.

- **C/C++:** The *sonar-cxx* plugin parses source text without rebuilding each historical project. It constructs a representation of the code's syntax and measures features such as branching complexity and duplication. The available checks depend on the plugin's parsing and rule coverage.
- **Java:** We point `sonar.java.binaries` and `sonar.java.libraries` to an empty directory when compiled class files and dependency libraries are unavailable. This permits source analysis without reconstructing the historical build, but limits checks that require compiled information.
- **JavaScript/TypeScript:** We allow up to 128 GB of heap memory through `sonar.javascript.node.maxspace=131072` and a one-hour analysis timeout through `sonar.javascript.analysisTimeout=3600`. The historical configuration disables bundle detection and the HTML sensor to reduce failures on large web applications.
- **Ruby:** We increase the Java virtual machine's stack allocation to 16 MB through `SONAR_SCANNER_OPTS=-Xss16m` to accommodate deeply nested code structures.
- **Rust:** We disable the optional Clippy integration through `sonar.rust.clippy.enable=false` because it requires an external toolchain. The source-code analyzer remains active.

C Stacked Difference-in-Differences

Design. The objective is to estimate how first institutional VC entry changes engineering quality when firms receive financing at different dates. Standard two-way fixed-effects regressions can use already-treated firms as controls, producing misleading estimates when treatment effects vary across cohorts or evolve over time. We therefore use the cohort-stacking approach in Gormley and Matsa (2011), constructing a separate comparison group that remains untreated throughout each cohort's event window. This appendix defines the cohort-event causal contrasts, states the assumptions under which they are identified, and relates them to the pooled regression used in the paper.

Cohorts and comparison firms. Let E_i denote the calendar month of firm i's first completed Seed round, or its first completed Series A round if no Seed is recorded. Set $E_i = \infty$ for firms without a qualifying financing. We exclude firms with a later-stage financing but no observed Seed or Series A history because their first institutional round cannot be dated. Following the cohort-stacking approach in Gormley and Matsa (2011), we compare each financing cohort with firms that remain untreated throughout its event window.

For cohort month c, define the treated and eligible control sets as

$$\mathcal{T}_c = \{i : E_i = c\}, \qquad \mathcal{J}_c = \{j : E_j > c + L\},$$

where $L = 12$ is the event-window half-width. Event time is $e = t - c$, with $e \in [-L, L]$; treatment begins at $e = 0$. The control set includes never-funded firms and firms funded after the window closes.

We stack the available observations for $\mathcal{T}_c \cup \mathcal{J}_c$ over the event window. At construction, a cohort must contain at least one treated observation at every observed event month and an eligible control group. Formally, let $\mathcal{E}_c$ denote the observed event months and let $n^0_{T,c,e}$ and $n^0_{C,c,e}$ count treated and control observations before estimation-sample filters. The construction rule retains

$$C^0_L = \left\{ c : \min_{e \in \mathcal{E}_c} n^0_{T,c,e} \geq 1, \quad \sum_{e \in \mathcal{E}_c} n^0_{C,c,e} \geq 1 \right\}.$$

After each estimation-sample restriction, we drop any cohort left without both treated and control firms; $C_L \subseteq C^0_L$ denotes the retained cohorts. The sample is unbalanced because each firm need not have a recorded scan in every month. A firm can also appear as a control in several earlier cohorts before entering its own treated cohort, which is why the stacked regression count exceeds the number of underlying firm–months.

Potential outcomes and identification. Let Y^g_{it} denote the outcome for firm i in calendar month t if its first financing occurs in month g, and let Y^∞_{it} denote the outcome without financing. The observed

outcome is $Y_{it} = Y_{it}^{E_i}$. This notation assumes that a firm's outcome is unaffected by the financing of other sample firms. The cohort-event average treatment effect on the treated is

$$\theta_{c,e} = \mathbb{E}\left[Y_{i,c+e}^{c} - Y_{i,c+e}^{\infty} \mid E_i = c\right].$$

A causal interpretation requires the following assumptions for the firms represented in each comparison.

Assumption PT (Parallel Trends). Absent financing, treated and control firms would have the same expected outcome change. For each post-treatment month $e \geq 0$ and pre-treatment reference month $\bar{e} < 0$,

$$\mathbb{E}\left[Y_{i,c+e}^{\infty} - Y_{i,c+\bar{e}}^{\infty} \mid E_i = c\right] = \mathbb{E}\left[Y_{i,c+e}^{\infty} - Y_{i,c+\bar{e}}^{\infty} \mid i \in \mathcal{J}_c\right].$$

Assumption NA (No Anticipation). Financing does not affect outcomes before the recorded event month:

$$Y_{it}^{g} = Y_{it}^{\infty} \quad \text{for } t < g.$$

We examine sensitivity to possible deal-adjacent changes, including fundraising preparation, using the donut exclusions in Section 4.4.

Under PT and NA, the population difference-in-differences contrast identifies $\theta_{c,e}$. To see this, write the observed change for treated firms and subtract the change for their cohort controls:

$$\begin{aligned}
\theta_{c,e}^{\mathrm{DID}} &= \mathbb{E}[Y_{i,c+e} - Y_{i,c+\bar{e}} \mid E_i = c] - \mathbb{E}[Y_{i,c+e} - Y_{i,c+\bar{e}} \mid i \in \mathcal{J}_c] \\
&= \mathbb{E}[Y_{i,c+e}^{c} - Y_{i,c+\bar{e}}^{\infty} \mid E_i = c] - \mathbb{E}[Y_{i,c+e}^{\infty} - Y_{i,c+\bar{e}}^{\infty} \mid i \in \mathcal{J}_c] \\
&= \mathbb{E}[Y_{i,c+e}^{c} - Y_{i,c+e}^{\infty} \mid E_i = c] \\
&\quad + \mathbb{E}[Y_{i,c+e}^{\infty} - Y_{i,c+\bar{e}}^{\infty} \mid E_i = c] - \mathbb{E}[Y_{i,c+e}^{\infty} - Y_{i,c+\bar{e}}^{\infty} \mid i \in \mathcal{J}_c] \\
&= \theta_{c,e}.
\end{aligned}$$

The second line uses no anticipation and the fact that controls remain untreated throughout the window. The third adds and subtracts the treated firms' counterfactual post-financing outcome, separating their treatment effect from the difference in untreated changes. Parallel trends makes the latter difference zero. Applied to the unbalanced estimation sample, this argument requires that the observation process preserves the counterfactual trends in PT; changing scan coverage must not itself produce a differential outcome change.

Aggregation and regression weights. A treated-firm-weighted summary of the cohort-event effects would be

$$\tau_e = \sum_{c \in C_L} \omega_c \theta_{c,e}, \qquad \omega_c = \frac{|\mathcal{T}_c|}{\sum_{c' \in C_L} |\mathcal{T}_{c'}|}, \qquad \tau_{\text{post}} = \frac{1}{L+1} \sum_{e=0}^{L} \tau_e,$$

where C_L denotes the retained cohorts. This aggregation weights cohorts by treated-firm counts and post-financing months equally. The pooled least-squares specification below uses weights determined by the available comparisons.

In the stacked estimation sample, define $D_{ice} = \mathbf{1}\{E_i = c\}\mathbf{1}\{e \geq 0\}$. The baseline specification is

$$Y_{ice} = \beta D_{ice} + \alpha_{ic} + \lambda_{ce} + \varepsilon_{ice},$$

Here Y_{ice} corresponds to $M_{i,c,t}$ in Equation (7), with $t = c + e$; D_{ice} corresponds to $VC_{i,c,t}$, and α_{ic} and λ_{ce} correspond to $\chi_{c,i}$ and $\delta_{c,t}$. The fixed effects absorb firm differences and common monthly shocks separately within each cohort. Let $\widetilde{D}_{ice}$ be the residual from projecting D_{ice} on these fixed effects, and let $\mathcal{O}$ denote the observed estimation rows. The estimated coefficient satisfies

$$\widehat{\beta} = \frac{\sum_{(i,c,e) \in \mathcal{O}} \widetilde{D}_{ice} Y_{ice}}{\sum_{(i,c,e) \in \mathcal{O}} \widetilde{D}_{ice}^2}.$$

The pooled coefficient combines the cohort comparisons according to their sample sizes and residual treatment variation. With balanced cohorts and equal window lengths, cohort weights are proportional to $|\mathcal{T}_c||\mathcal{J}_c|/(|\mathcal{T}_c| + |\mathcal{J}_c|)$. In the unbalanced sample, the pattern of observed firm–months also affects the weights, which need not be positive for every underlying effect or coincide with the explicit treated-firm weights in τ_{post}. We report $\widehat{\beta}$ as the pooled regression contrast defined above, using cohort construction to avoid already-treated controls. Standard errors are clustered by cohort and firm to account for shared cohort shocks and the repeated appearance of firms across stacks. Appendix N examines sensitivity to control reuse.

Event-study implementation. The quarterly event study replaces D_{ice} with interactions between cohort treatment status and event-quarter indicators:

$$Y_{ice} = \sum_{q \in Q,\, q \neq 0} \beta_q \mathbf{1}\{E_i = c\}\mathbf{1}\{q(e) = q\} + \alpha_{ic} + \lambda_{ce} + \varepsilon_{ice},$$

where Q is the set of observed event-quarter bins. The omitted bin $q = 0$ contains months $\{-4, -3, -2\}$. The next bin, $q = 1$, contains months $\{-1, 0, 1\}$ and combines one pre-treatment month with two

months at or after financing. It therefore averages over the onset of treatment; later bins are entirely post-treatment. The coefficients are normalized relative to the omitted pre-treatment bin and inherit the pooled regression's weighting. The pre-treatment coefficients provide an observable diagnostic for differential trends before financing. Under parallel pre-treatment paths and no anticipation, they should fluctuate around zero; the plots and joint tests assess this prediction before tracing the post-financing response.

D Matching on Pre-Treatment Headcount Growth

We match within financing cohorts on growth measured before the round. For firm i in cohort c, let $H_{i,c,e}$ denote headcount at event month e. The matching feature is

$$G_i^c = \ln\left(\frac{H_{i,c,-4} + H_{i,c,-3} + H_{i,c,-2}}{3}\right) - \ln\left(\frac{H_{i,c,-12} + H_{i,c,-11}}{2}\right).$$

This feature compares average headcount near the beginning and end of the pre-financing period, excluding the month immediately before the round to reduce sensitivity to deal-adjacent hiring. Sensitivity specifications add analogously constructed growth in software-engineering headcount and lines of code, making G_i^c a vector.

Using the notation of Appendix C, the candidate controls satisfy $E_j > c + 12$. We require an observation at $e = -1$ and nonmissing matching features for both treated firms and candidate controls. Let $\bar{G}_c^{\mathcal{T}}$ be the mean feature vector among eligible treated firms. In the primary matched specification, the distance between a candidate control and this treated-group mean is

$$d_c(j) = \sqrt{(G_j^c - \bar{G}_c^{\mathcal{T}})' \widehat{\Sigma}_c^{-1} (G_j^c - \bar{G}_c^{\mathcal{T}})},$$

where $\widehat{\Sigma}_c$ is the sample covariance of the candidate-control vectors together with the treated-group mean. With headcount growth as the sole feature, this is the absolute growth difference divided by the corresponding sample standard deviation.

We retain controls with $d_c(j) \leq 1$, ranked by distance, and keep the available event-window observations for eligible treated firms and selected controls. Each control enters at most once within a cohort but can appear in several cohorts, and we discard any cohort left without both groups.[13] Matching to the treated-group mean constructs a comparison group resembling the typical newly funded startup in the cohort. It therefore addresses the observed difference in growth trajectories while retaining more eligible controls than individual one-to-one matching.

Appendix E examines whether this result depends on the particular definition of a comparable

[13] The implementation caps the number of selected controls at 1,000 times the number of treated firms in the cohort. This cap does not bind, so the distance cutoff determines selection.

firm. We vary the distance cutoff from 0.5 to 2.0, add growth in code volume and software-engineering headcount, and replace matching to the treated-group mean with individual nearest-neighbor matching at $K \in \{1, 3\}$, selecting controls sequentially without replacement within a cohort. These alternatives change both the dimensions along which firms are matched and how tightly their trajectories must align, allowing us to assess whether the quality improvement persists across plausible comparison groups.

E Matching Sensitivity Analysis

This appendix examines the sensitivity of the matched difference-in-differences estimates to the choice of matching configuration. We construct a specification curve (Simonsohn, Simmons, and Nelson, 2020) over 72 configurations that vary along three dimensions: **(i)** the feature set used to compute Mahalanobis distances (headcount growth only; headcount plus lines of code; headcount plus software engineering headcount; all three); **(ii)** the matching mode (centroid matching, individual nearest-neighbor with $K = 1$, and individual nearest-neighbor with $K = 3$); and **(iii)** the caliper, measured in Mahalanobis-distance units ($\delta \in \{0.5, 0.75, 1.0, 1.25, 1.5, 2.0\}$). We match without replacement within each cohort; controls may appear across different cohorts but are used at most once per cohort. For each configuration, we re-estimate the stacked difference-in-differences specification in Equation (7) on the resulting matched sample.

Figures 16–19 present the specification curves for our four core engineering-quality outcomes. In each figure, the top panel plots the point estimate and 95% confidence interval for every specification, grouped by feature set, matching mode, and caliper in the same order across outcomes. The dashed gray line marks the baseline (unmatched) estimate and its 95% confidence band (gray shading); the solid gray line marks zero. The orange diamond identifies our primary matched specification (centroid matching on headcount growth, caliper $\delta = 1$). The bottom panel displays an indicator matrix identifying the feature set, matching mode, and caliper active in each specification.

The direction of the quality estimates is stable across matching choices. Every critical-violation, total-violation, and maintainability-issue coefficient is negative, as are 68 of the 72 technical-debt coefficients. At the 5% level, 26% of critical-violation specifications, 61% of total-violation specifications, and 61% of maintainability-issue specifications reject the null; the corresponding shares at the 10% level are 85%, 74%, and 79%. No technical-debt specification reaches the 5% level, although six reach the 10% level.

Matching substantially reduces the sample: the primary specification retains approximately 72,000 of 488,000 baseline observations, and some individual nearest-neighbor configurations retain fewer than 10,000. Despite this contraction, the median matched-to-baseline coefficient ratio is 1.38 for critical violations, 0.90 for total violations, and 0.86 for maintainability issues. The stability of these magnitudes, alongside wider confidence intervals, indicates that reduced statistical power

contributes to the loss of significance. By contrast, technical debt is less stable, with a median ratio of 0.63 and no estimate significant at the 5% level. The primary matched specification lies near the median of the coefficient distribution for all four outcomes, showing that its results are representative of the broader matching exercise rather than an unusual choice of matching parameters.

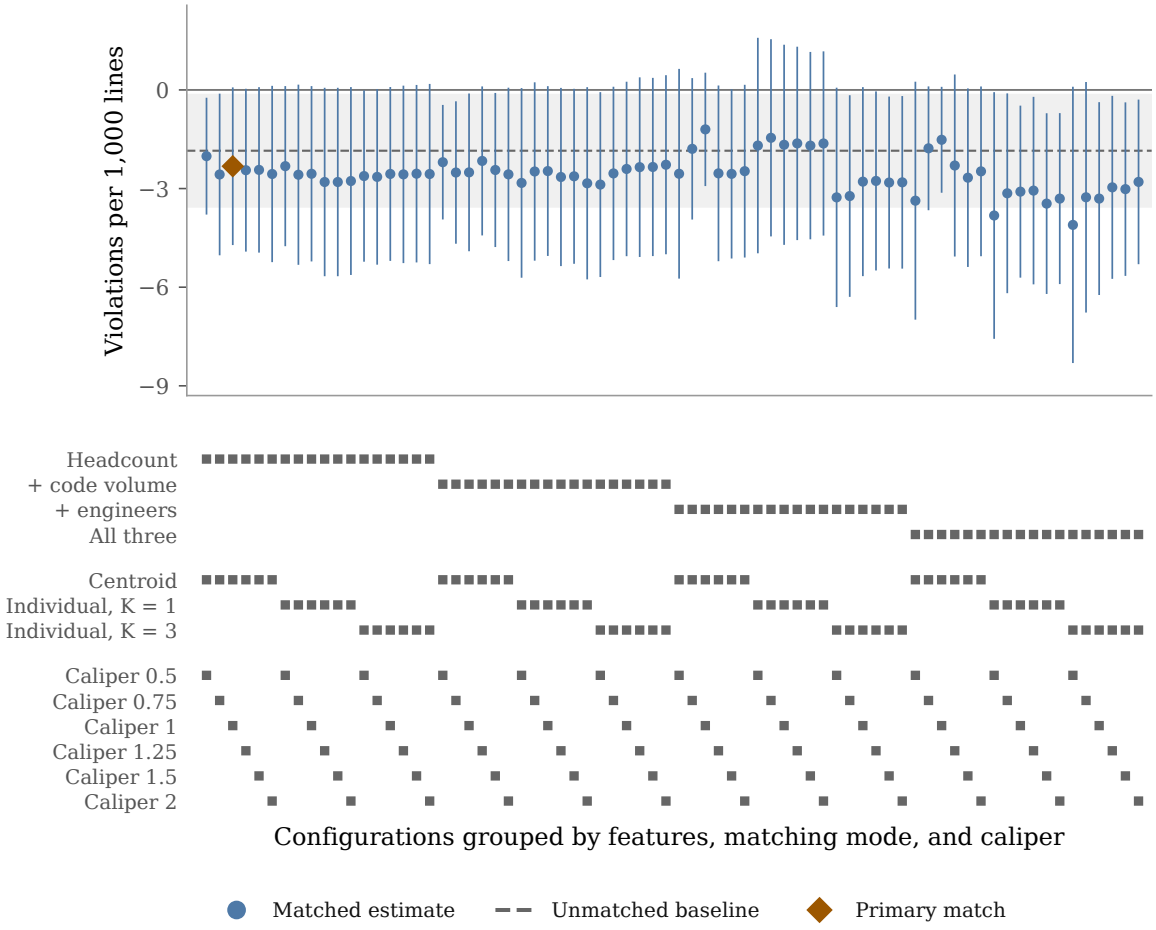


Figure 16: Specification Curve: Critical Violations per 1,000 LoC. Estimates across 72 matching configurations, identified in the lower panel. The dashed line and shaded band show the unmatched estimate and its interval; the orange diamond marks the preferred matched specification. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

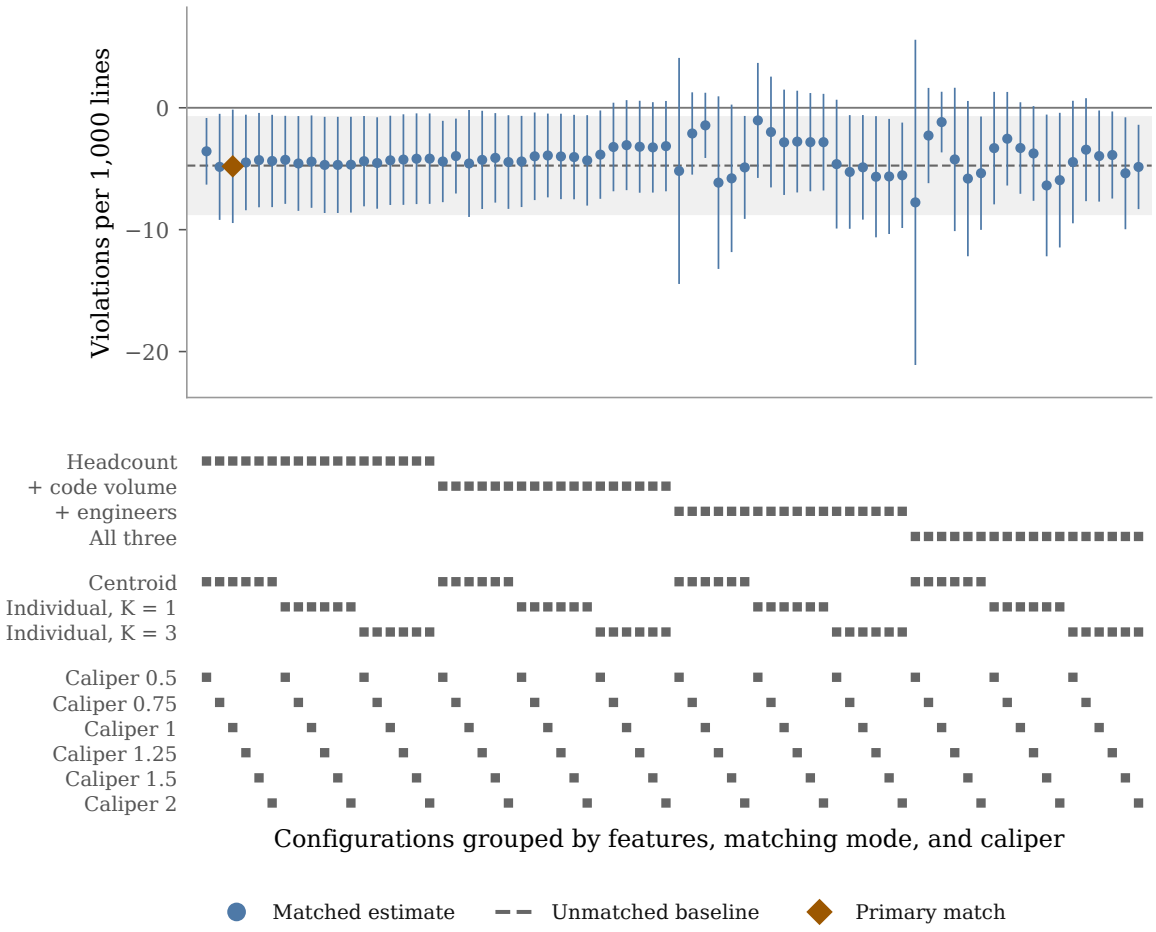


Figure 17: Specification Curve: Violations (All) per 1,000 LoC. See Figure 16 for a description of the layout.

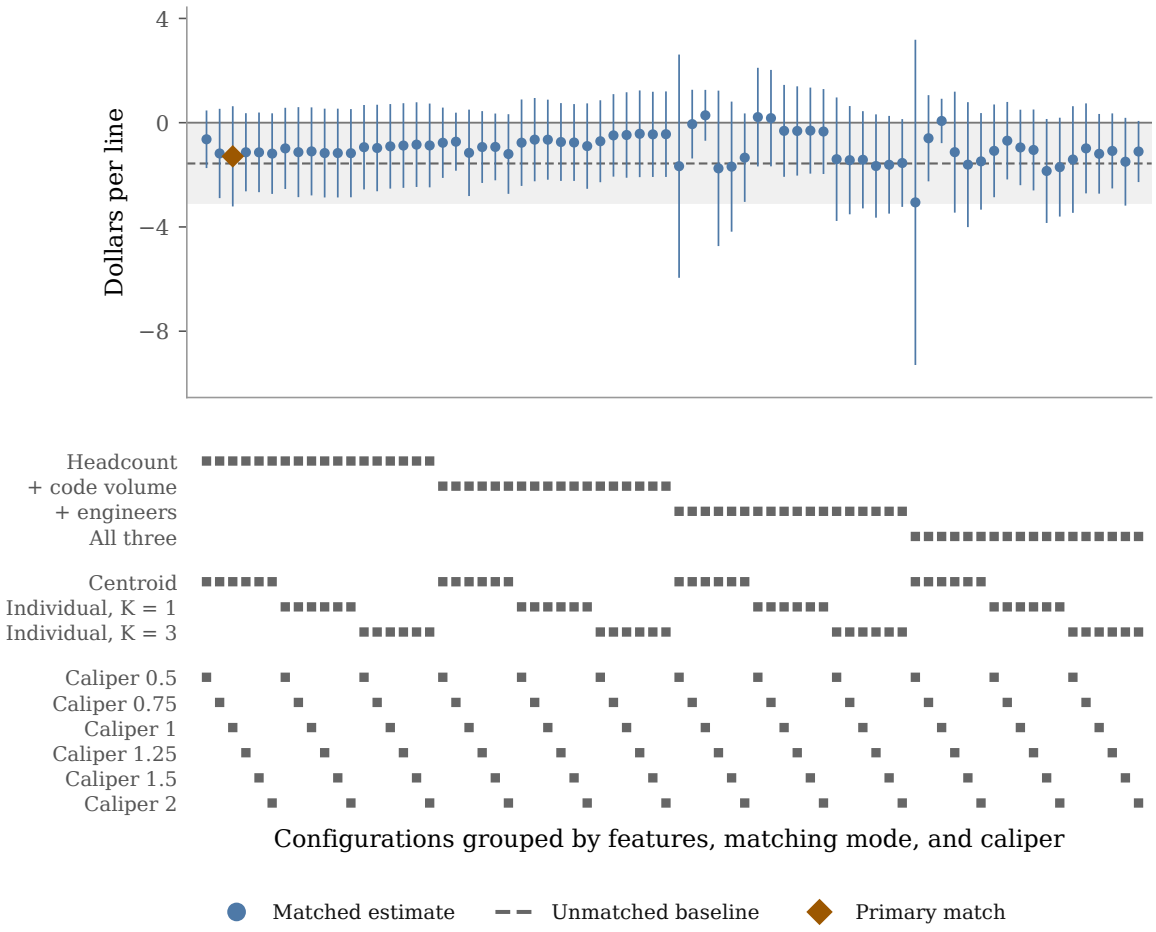


Figure 18: Specification Curve: Technical Debt per Line of Code. See Figure 16 for a description of the layout.

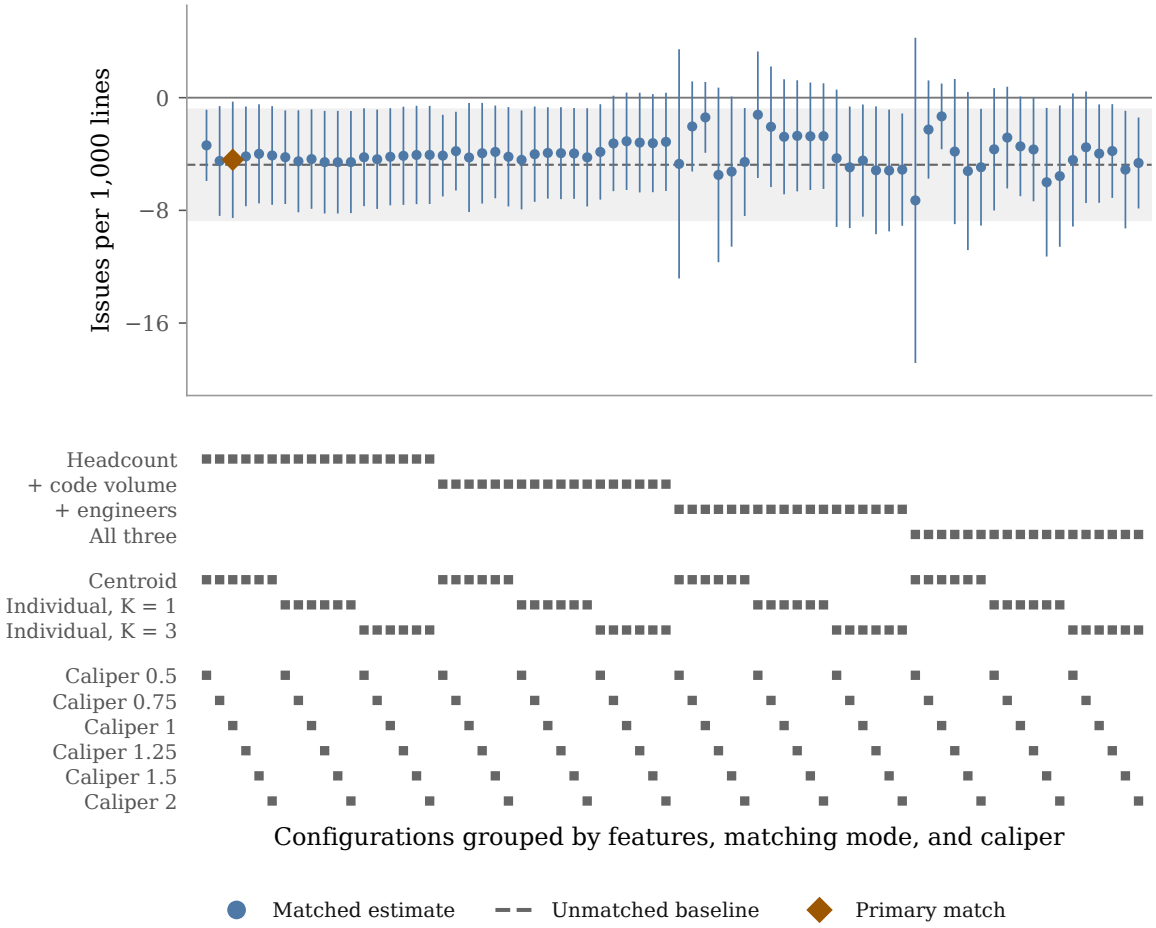


Figure 19: Specification Curve: Maintainability Issues per 1,000 LoC. See Figure 16 for a description of the layout.

F Founding-Team Heterogeneity

We examine whether the engineering response depends on the founding team's technical expertise and prior entrepreneurial experience. If VC involvement primarily supplies knowledge that founders lack, firms with technically experienced or repeat founders should have less scope to improve. A broader change in organizational practices could instead reach firms with either type of founding team. Using LinkedIn career histories, we identify individuals whose titles contain "founder" or "co-founder" and who were active at the firm within six months of first VC entry, then retrieve their prior employment records to construct two indicators.

Technical Founder equals one if any member of the founding team simultaneously held a technical leadership role—CTO, VP of Engineering, Head of Engineering, or Director of Engineering—at the

same firm around the deal date. Such founders may have established engineering practices before financing, leaving less scope for subsequent change.

Serial Entrepreneur equals one if any founder previously held a CEO, Chief, Founder, or President role at a different company before the first VC deal date. Serial entrepreneurs may bring established engineering practices and organizational templates from prior ventures, which could reduce the scope for further changes after financing.

We interact each indicator with the Treated $\times$ Post term, following the same triple-interaction specification used for the pre-treatment quality and deal size splits in the main analysis. We merge the founder indicators from LinkedIn onto the main panel; the founder columns cover firms with the required career histories. Tables 20–22 report the full regression output. Each table presents eight columns: columns (1)–(4) correspond to one outcome, and columns (5)–(8) to a second related outcome, with each set of four columns cycling through the Large Codebase, Statically Typed Language, Technical Founder, and Serial Entrepreneur interaction specifications in that order.

Table 20: Founder Heterogeneity: Technical Debt per LoC and Technical Debt Ratio

	Technical Debt per LoC				**Technical Debt Ratio**			
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Treated × Post	−2.136	−2.946**	−1.075	−1.255	−7.766	−10.713**	−3.911	−4.565
	(1.306)	(1.232)	(1.109)	(1.590)	(4.748)	(4.480)	(4.034)	(5.782)
Post × Large Codebase (=1)	0.087				0.317			
	(0.415)				(1.509)			
Treated × Post × Large Codebase (=1)	1.287				4.681			
	(1.371)				(4.985)			
Post × Static Typed (=1)		0.023				0.085		
		(0.298)				(1.084)		
Treated × Post × Static Typed (=1)		3.116**				11.330**		
		(1.212)				(4.408)		
Post × Technical Founder (=1)			0.055				0.200	
			(0.753)				(2.740)	
Treated × Post × Technical Founder (=1)			0.205				0.744	
			(1.773)				(6.448)	
Post × Serial Entrepreneur (=1)				0.384				1.397
				(0.852)				(3.098)
Treated × Post × Serial Entrepreneur (=1)				0.500				1.818
				(1.824)				(6.631)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	477,107	487,577	78,620	78,620	477,107	487,577	78,620	78,620
R^2	0.828	0.828	0.783	0.783	0.828	0.828	0.783	0.783

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$. Technical Debt per LoC is calibrated remediation cost in dollars per line; the Technical Debt Ratio is that debt as a percentage of replacement-cost software assets. Columns (1) and (5) interact with a Large Codebase indicator (above-median lines of code). Columns (2) and (6) interact with a Statically Typed Language indicator. Columns (3) and (7) interact with a Technical Founder indicator (founding team includes a simultaneous CTO/VP Eng/Director Eng role). Columns (4) and (8) interact with a Serial Entrepreneur indicator (any founder previously held a C-suite or founder role at a prior firm). See Appendix A, Table 19, for full variable definitions.

The employment-based indicators are available only for firms with LinkedIn coverage; those columns run on the covered subsample rather than coding missing markers as zero, so Columns (3), (4), (7), and (8) have substantially smaller samples than Columns (1), (2), (5), and (6).

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

Table 21: Founder Heterogeneity: Violations per kLoC and Critical Violations per kLoC

	Violations per kLoC				**Critical Violations per kLoC**			
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Treated × Post	−6.347*	−8.714**	−2.934	−3.629	−2.354	−3.567**	−2.075	−2.809
	(3.418)	(3.358)	(2.202)	(2.815)	(1.487)	(1.407)	(1.622)	(2.204)
Post × Large Codebase (=1)	1.635**				0.532			
	(0.806)				(0.422)			
Treated × Post × Large Codebase (=1)	4.161				1.303			
	(3.538)				(1.587)			
Post × Static Typed (=1)		−0.226				0.251		
		(0.677)				(0.321)		
Treated × Post × Static Typed (=1)		9.058***				3.804***		
		(3.329)				(1.371)		
Post × Technical Founder (=1)			−0.309				0.107	
			(1.797)				(0.840)	
Treated × Post × Technical Founder (=1)			−1.091				0.390	
			(4.174)				(2.746)	
Post × Serial Entrepreneur (=1)				1.510				−0.595
				(1.851)				(0.891)
Treated × Post × Serial Entrepreneur (=1)				0.697				1.540
				(3.599)				(2.761)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	477,107	487,577	78,620	78,620	477,107	487,577	78,620	78,620
R^2	0.857	0.855	0.826	0.826	0.885	0.881	0.814	0.814

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$. Columns (1) and (5) interact with a Large Codebase indicator. Columns (2) and (6) interact with a Statically Typed Language indicator. Columns (3) and (7) interact with a Technical Founder indicator. Columns (4) and (8) interact with a Serial Entrepreneur indicator. Violations per kLoC counts total rule violations across all severity levels per 1,000 lines of code. Critical Violations per kLoC counts violations classified as critical per 1,000 lines of code. See Appendix A, Table 19, for full variable definitions.

The employment-based indicators are available only for firms with LinkedIn coverage; those columns run on the covered subsample rather than coding missing markers as zero, so Columns (3), (4), (7), and (8) have substantially smaller samples than Columns (1), (2), (5), and (6).

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

Table 22: Founder Heterogeneity: Bugs per kLoC and Reliability Issues (High) per kLoC

	Bugs per kLoC				**Reliability Issues (High) per kLoC**			
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Treated × Post	0.041	−0.137	0.052	−0.036	−0.010	−0.001	0.006	−0.029
	(0.198)	(0.170)	(0.156)	(0.136)	(0.029)	(0.025)	(0.018)	(0.022)
Post × Large Codebase (=1)	0.082				−0.022			
	(0.081)				(0.018)			
Treated × Post × Large Codebase (=1)	−0.079				0.001			
	(0.240)				(0.029)			
Post × Static Typed (=1)		0.139				−0.014		
		(0.084)				(0.010)		
Treated × Post × Static Typed (=1)		0.249				−0.006		
		(0.190)				(0.024)		
Post × Technical Founder (=1)			−0.058				0.016	
			(0.289)				(0.012)	
Treated × Post × Technical Founder (=1)			−0.677				−0.021	
			(0.457)				(0.025)	
Post × Serial Entrepreneur (=1)				0.334				−0.002
				(0.286)				(0.013)
Treated × Post × Serial Entrepreneur (=1)				−0.247				0.052*
				(0.366)				(0.030)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	477,107	487,577	78,620	78,620	477,107	487,577	78,620	78,620
R^2	0.904	0.903	0.917	0.917	0.507	0.505	0.752	0.752

This table reports the coefficient β from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$. Columns (1) and (5) interact with a Large Codebase indicator. Columns (2) and (6) interact with a Statically Typed Language indicator. Columns (3) and (7) interact with a Technical Founder indicator. Columns (4) and (8) interact with a Serial Entrepreneur indicator. Bugs per kLoC counts patterns classified by SonarQube as likely to cause incorrect or unexpected behavior per 1,000 lines of code. Reliability Issues (High) per kLoC counts high-severity reliability defects posing significant risk to system stability per 1,000 lines of code. See Appendix A, Table 19, for full variable definitions.

The employment-based indicators are available only for firms with LinkedIn coverage; those columns run on the covered subsample rather than coding missing markers as zero, so Columns (3), (4), (7), and (8) have substantially smaller samples than Columns (1), (2), (5), and (6).

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

Interpretation. The Technical Founder interactions are statistically insignificant across the six outcomes. The Serial Entrepreneur interaction is also insignificant except for high-severity reliability

issues, where it is 0.052 per 1,000 lines (SE = 0.030, $p < 0.10$). For technical debt per line, the Technical Founder interaction is \$0.205 (SE = 1.773), and the Serial Entrepreneur interaction is \$0.500 (SE = 1.824). Thus, neither characteristic shows a systematic association with the engineering response, with only one marginally significant coefficient among twelve interaction tests.

This is informative about the prediction that VC professionalization mainly compensates for a lack of founder expertise. The quality gains are not systematically concentrated among firms whose founders lack technical or entrepreneurial experience. The pattern is consistent with organizational changes—such as formal review procedures and coordination across developers—that can benefit technically sophisticated founders as well as less experienced teams. We interpret the absence of a systematic gradient as evidence about this prediction, while the standard errors limit what can be concluded about equality across founder groups.

Post-Funding Hiring Composition

We next examine the complementary human-capital prediction that firms recruiting senior technical talent should improve more after financing. *Hired Technical Leader* equals one if the firm hires a CTO, VP of Engineering, or Head of Engineering within 24 months of its first round. *High Senior Ratio* equals one if senior engineering hires as a share of all new hires over that window exceed the treated-firm median. We interact each indicator with Treated × Post, using post-financing hiring to assess where the observed quality gains are concentrated.

Tables 23–24 report the estimates. For each outcome, the first column uses the technical-leader indicator and the second uses the senior-hire ratio. We exclude missing hiring measures rather than coding them as zero. The available employment records restrict these specifications to 68,498 stacked observations, compared with 487,577 in the main analysis.

Table 23: Hiring-Composition Heterogeneity: Critical Violations and Total Violations per kLoC

	Critical Violations / kLoC		**Violations / kLoC**	
	(1)	(2)	(3)	(4)
Treated × Post	−5.123	−3.354	−9.624**	−5.286
	(3.918)	(3.008)	(4.774)	(3.761)
Post × Hired Technical Leader	−0.894		−3.859	
	(1.089)		(2.678)	
Treated × Post × Hired Technical Leader	4.674		10.411*	
	(3.938)		(5.370)	
Post × High Senior Ratio		0.530		−0.222
		(0.956)		(2.007)
Treated × Post × High Senior Ratio		2.025		3.423
		(2.981)		(4.292)
Cohort × Firm FE	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes
# Obs	68,498	68,498	68,498	68,498
R^2	0.815	0.815	0.828	0.827

This table reports triple-interaction specifications comparing post-financing quality changes across hiring groups. Hired Technical Leader is an indicator equal to one if the firm hired a CTO, VP of Engineering, or Head of Engineering within 24 months of funding. High Senior Ratio is an indicator equal to one if senior engineering hires as a share of all new hires exceed the treated-firm median. Column (1) introduces the Hired Technical Leader interaction and Column (2) the High Senior Ratio interaction for Critical Violations; Columns (3) and (4) do the same for Violations. Critical Violations and Violations are measured per 1,000 lines of code.

The employment-based indicators are available only for firms with LinkedIn coverage; those columns run on the covered subsample rather than coding missing markers as zero.

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

Table 24: Hiring-Composition Heterogeneity: Technical Debt per LoC and Maintainability Issues per kLoC

	TD per LoC ($)		**Maint Issues / kLoC**	
	(1)	(2)	(3)	(4)
Treated × Post	−3.134	−1.862	−8.751*	−4.846
	(2.217)	(1.683)	(4.530)	(3.580)
Post × Hired Technical Leader	−1.122		−3.192	
	(1.273)		(2.398)	
Treated × Post × Hired Technical Leader	3.319		9.255*	
	(2.577)		(5.013)	
Post × High Senior Ratio		−0.291		0.140
		(0.877)		(1.865)
Treated × Post × High Senior Ratio		1.391		2.949
		(1.983)		(4.044)
Cohort × Firm FE	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes
# Obs	68,498	68,498	68,498	68,498
R^2	0.783	0.783	0.816	0.816

This table reports triple-interaction specifications comparing post-financing quality changes across hiring groups. Hired Technical Leader is an indicator equal to one if the firm hired a CTO, VP of Engineering, or Head of Engineering within 24 months of funding. High Senior Ratio is an indicator equal to one if senior engineering hires as a share of all new hires exceed the treated-firm median. Column (1) introduces the Hired Technical Leader interaction and Column (2) the High Senior Ratio interaction for Technical Debt per LoC; Columns (3) and (4) do the same for Maintainability Issues. Technical Debt per LoC is measured in dollars. Maintainability Issues are measured per 1,000 lines of code.

The employment-based indicators are available only for firms with LinkedIn coverage; those columns run on the covered subsample rather than coding missing markers as zero.

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

The High Senior Ratio interaction is statistically insignificant for all four outcomes. The Hired Technical Leader interaction is positive but insignificant for critical violations (+4.674, SE = 3.938) and technical debt per line (+3.319, SE = 2.577). It reaches the 10% level for total violations (+10.411, SE = 5.370) and maintainability issues (+9.255, SE = 5.013), indicating smaller estimated declines among firms that hire a technical leader. The negative base estimates remain statistically significant in

every hiring specification. Together with the founder comparisons, these results provide little support for improvements driven principally by the addition of senior technical expertise and instead motivate the examination of existing teams in Section 3.8.

G Defect Levels versus Defect Density

The main results concern the quality of software production, measured by defects per unit of code. This normalization is analogous to a manufacturing defect rate: a producer can make more units and accumulate more defects in total while reducing the share of output that is defective. In software, each addition or modification to a codebase can introduce a new issue, so defect density helps distinguish the quality of engineering practices from the amount of code produced. We examine that distinction by estimating the baseline specification on total critical violations, total violations, and total technical debt, each transformed as $\log(1+x)$.

Table 25 compares the density and stock estimates. Critical violations fall by 1.85 per 1,000 lines of code in Column (1), significant at the 5% level, while the transformed critical-violation count has a positive but statistically insignificant coefficient of 0.075 (SE = 0.069) in Column (3). Technical debt falls by $1.57 per line in Column (2), but its transformed dollar stock rises by 0.196 (SE = 0.077) in Column (4); both estimates are significant at the 5% level. Total violations also increase in the log specification (0.135, SE = 0.064, significant at the 5% level; not tabulated). These findings show why counts and densities answer different economic questions: the accumulated burden can increase as the product grows even when each unit of software carries less debt.

Table 25: Defect and Technical Debt Outcomes: Densities and Stocks

	Density		**Stocks** ($\log(1+x)$)	
	Critical Violations / kLoC	TD per LoC ($)	Critical Violations	TD ($)
	(1)	(2)	(3)	(4)
Treated × Post	−1.849**	−1.566**	0.075	0.196**
	(0.863)	(0.765)	(0.069)	(0.077)
Cohort × Firm FE	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes
# Obs	487,577	487,577	487,577	487,577
R^2	0.881	0.828	0.922	0.902

This table compares the stacked DiD estimates for defect and technical debt densities with estimates for the corresponding stocks. Columns (1)–(2) reproduce the baseline per-LoC results from Tables 3 and 7. Columns (3)–(4) estimate the same specification using log(1 + critical violations) and log(1 + technical debt in dollars). All specifications include Firm × Cohort and Month × Cohort fixed effects. See Appendix A, Table 19, for full variable definitions.

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

A constant defect rate provides a useful benchmark for these magnitudes. If defect stocks expanded proportionately with code, their log growth would approximately match the code-volume coefficient of 0.226, corresponding to a 25.4% expansion relative to controls. The critical-violation, total-violation, and debt-stock coefficients all fall below that benchmark. The comparison is consistent with firms adding code to a higher standard as they expand, and it explains how significant density improvements can coexist with no statistically significant reduction in absolute critical-violation counts.[14]

The count-density comparison motivates two more direct checks on the role of codebase growth, reported in Appendix H. Appendix I presents the stock event studies, while the contribution-level analysis in Section 3.8 provides separate evidence that the output of incumbent engineers improves after financing.

H Decomposing the Quality Improvement

The decline in defect density survives two tests that address the role of codebase growth: it strengthens among firms with low pre-financing growth and remains statistically significant in a count model that explicitly controls for code volume. These exercises matter because density alone cannot distinguish a change in engineering practices from the arithmetic effect of adding code to the denominator. We present each test below, then use a log-density decomposition to relate the results to the stock estimates.

Pre-Treatment Codebase Growth

We first examine firms that exhibited low codebase growth before financing. We retain treated firms in the bottom quartile of growth in $\log(1 + \text{LoC})$ between months $[-12, -7]$ and $[-6, -1]$, together with controls in their cohorts. This classification uses only pre-treatment information. Pre- and post-treatment growth correlate at 0.57 among treated firm–cohorts, and 62% of observations in the lowest pre-growth quartile remain in the lowest post-growth quartile. The log-code-volume coefficient in this subsample is −0.016 (SE = 0.048, $p = 0.74$), compared with +0.226 in the full sample. The restriction therefore identifies a group with much less estimated post-financing code expansion.

Table 26 shows that the critical-violation estimate strengthens in this subsample. Critical violations decline by 2.852 per 1,000 lines of code (SE = 1.533, $p = 0.067$), compared with 1.849 in the full sample. Technical debt per line declines by \$1.639 ($p = 0.052$), and total violations by 3.720 per 1,000 lines ($p < 0.05$). Joint tests do not reject zero pre-treatment coefficients for these outcomes. Thus, the quality improvement persists in a group with little estimated post-financing code expansion, where rapid denominator growth is a less compelling explanation. A full-sample tercile interaction likewise finds no systematic gradient in the critical-violation estimate with pre-treatment growth (Wald $p = 0.20$ for the highest-versus-lowest tercile comparison).

[14] The comparison concerns estimated changes in $\log(1 + x)$ and is approximate, particularly for small counts. The scale outcomes have different pre-financing trajectories, and lower aggregate density can also reflect remediation, deletion, or changes in code composition.

The larger critical-violation estimate also has an economic interpretation. Firms whose codebases expand slowly may have more scope to devote engineering effort to existing software, including remediation and improvements in development procedures. Alternatively, the full-sample estimate may combine these gains with the additional coordination demands of rapid feature development. The low-growth result is consistent with both possibilities: improved engineering discipline need not depend on producing large amounts of additional code.

Table 26: Engineering Quality Among Firms with Low Pre-Treatment Codebase Growth

	Critical Violations / kLoC		**TD per LoC**	
	Full Sample	Low Pre-Treatment Growth (Q1)	Full Sample	Low Pre-Treatment Growth (Q1)
	(1)	(2)	(3)	(4)
Treated × Post	−1.849**	−2.852*	−1.566**	−1.639*
	(0.863)	(1.533)	(0.765)	(0.830)
Cohort × Firm FE	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes
# Obs	487,577	361,439	487,577	361,439

This table compares the baseline per-LoC estimates (Columns 1 and 3) with estimates from a subsample restricted to treated firms in the bottom quartile of *pre-treatment* codebase growth (log LoC growth from months $[-12,-7]$ to $[-6,-1]$), retaining all control firms in their cohorts (Columns 2 and 4). The restriction uses only pre-treatment information. In this subsample, the coefficient for $\log(1+\text{LoC})$ is -0.016 (SE = 0.048, $p = 0.74$); this null does not imply constant code volume. Cohorts left with only treated or only control firms after the restriction are dropped. Total violations per kLoC in the same subsample: -3.720 (SE = 1.852, $p < 0.05$). Joint tests do not reject zero pre-treatment coefficients for the reported outcomes. See Appendix A, Table 19, for variable definitions.
Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

Poisson Pseudo-Maximum Likelihood

We next estimate a Poisson pseudo-maximum-likelihood model for violation counts, including $\log(1+\text{LoC})$ as a control with a freely estimated coefficient. This specification models the number of defects directly and allows it to scale nonproportionally with codebase size. Because current code volume can respond to financing, the coefficient measures the quality change conditional on scale, complementing the baseline estimate of the total change in defect density.

Table 27: PPML Estimates for Violation Counts with a Code-Volume Control

Dependent variable:	**Critical Violations**	**Violations (All)**
	(1)	(2)
Treated × Post	−0.089**	−0.052*
	(0.044)	(0.030)
log(1 + LoC)	1.199***	1.147***
	(0.076)	(0.061)
Cohort × Firm FE	Yes	Yes
Month × Cohort FE	Yes	Yes
# Obs	458,195	485,507

This table reports Poisson pseudo-maximum-likelihood estimates with log(1 + LoC) as a freely estimated control for codebase size. The dependent variables are the raw counts of critical violations and total violations, not normalized per LoC. Exponentiating the Treated × Post coefficient gives the proportional change in the model's conditional mean: $\exp(-0.089) - 1 = -8.5\%$ for critical violations. The observation counts are lower than the baseline following PPML estimation exclusions. The single-repository subsample delivers nearly identical estimates (−0.093, $p = 0.02$, for critical violations).

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

Column (1) of Table 27 reports a coefficient of −0.089 (SE = 0.044), significant at the 5% level. The model implies 8.5% fewer critical violations after financing relative to controls, conditional on code volume; total violations are 5.1% lower ($p = 0.084$). The code-volume coefficient is 1.20 for critical violations in the main sample and 1.05 in the single-repository subsample. The former implies that a 1% increase in code volume is associated with approximately 1.2% more critical violations, conditional on the other regressors. Allowing for this relationship between product size and defects leaves a negative financing coefficient, supporting the quality result in a model that does not divide defects by code volume.

Log-Density Decomposition

We also estimate the baseline specification on log(1 + count) − log(1 + LoC), a transformed density measure that accommodates zero counts. The critical-violation coefficient is −0.151 ($p = 0.005$), and the total-violation coefficient is −0.091 ($p = 0.019$). The corresponding pre-treatment tests reject at

the 10% level for critical violations ($p = 0.096$) but do not reject for total violations, so the critical log-density specification has weaker pre-trend support than the baseline density measure.

The critical-violation estimate decomposes into a count coefficient of +0.075 ($p = 0.29$) and a code-volume coefficient of +0.226 ($p < 0.01$). The identity makes the source of the transformed density decline explicit: the estimated count change is smaller than the estimated change in code volume. The single-repository subsample yields negative density estimates of slightly larger magnitude. This decomposition describes the accounting relationship between the two components; the low-growth and Poisson exercises above test the result under alternative treatments of scale.

Both tests support an improvement in engineering practices. The low-growth subsample shows that the critical-violation decline persists, and increases in magnitude, among firms with little estimated code expansion. The Poisson model confirms a reduction in critical-violation counts after explicitly accounting for codebase size. The main finding therefore extends beyond the particular normalization used in the baseline regressions and is consistent with greater engineering discipline as firms develop their products.

I Level Outcomes: Event-Study Figures

This appendix presents event-study estimates for the stock of code issues and technical debt. Each dependent variable is the natural logarithm of one plus the relevant count or, for technical debt, one plus its calibrated dollar value. These measures describe changes in the total burden as the codebase grows; the per-line measures in the main text describe its density.

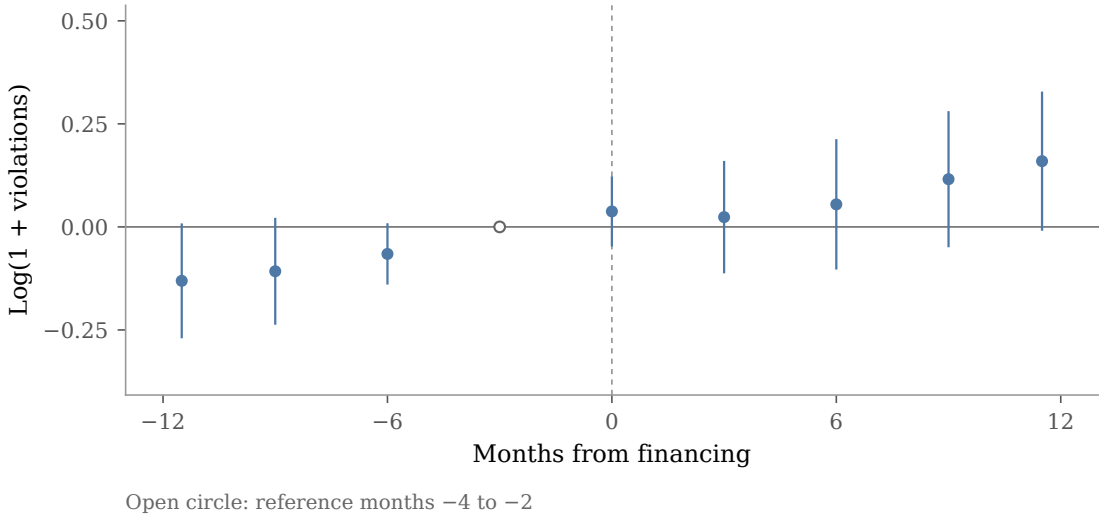


Figure 20: Log(Total Violations). Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

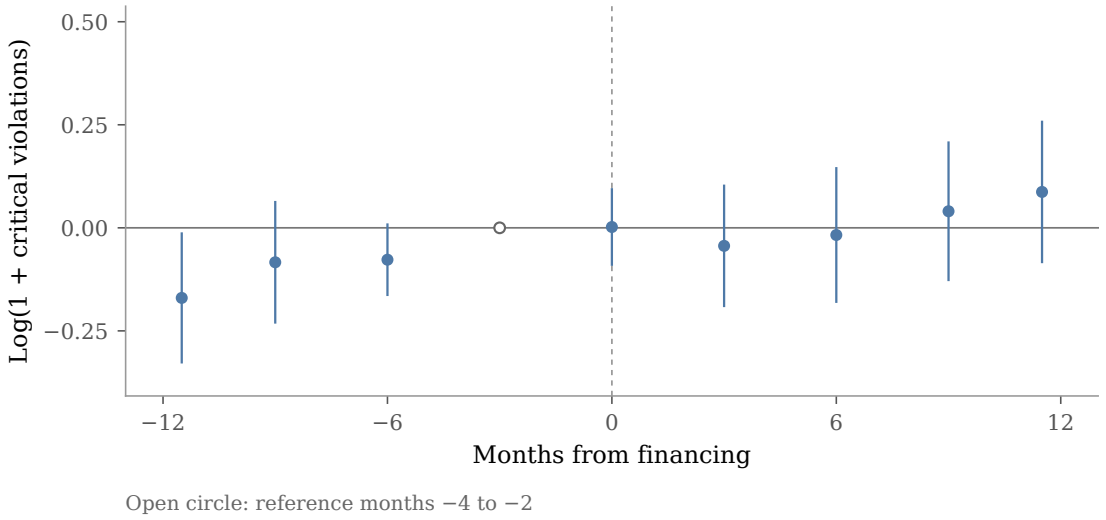


Figure 21: Log(Critical Violations). Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

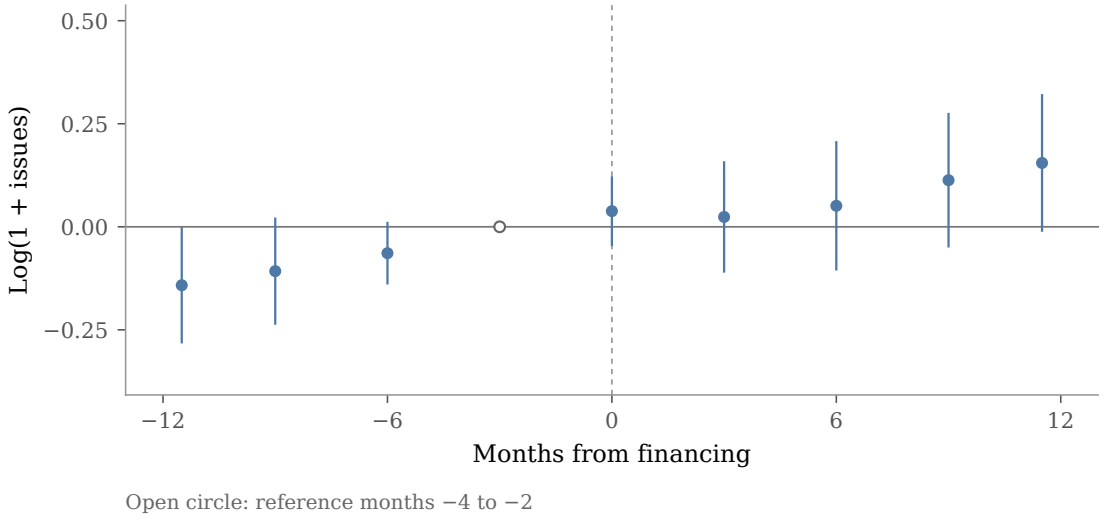


Figure 22: Log(Total Maintainability Issues). Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

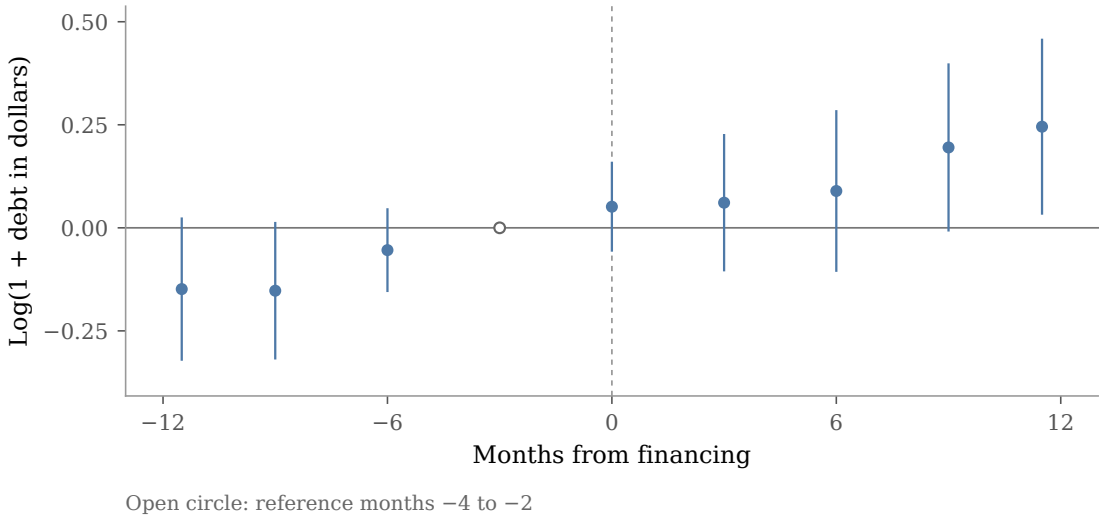


Figure 23: Log(Technical Debt $). Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

J Series B+ Placebo Estimates: Later Financing Rounds

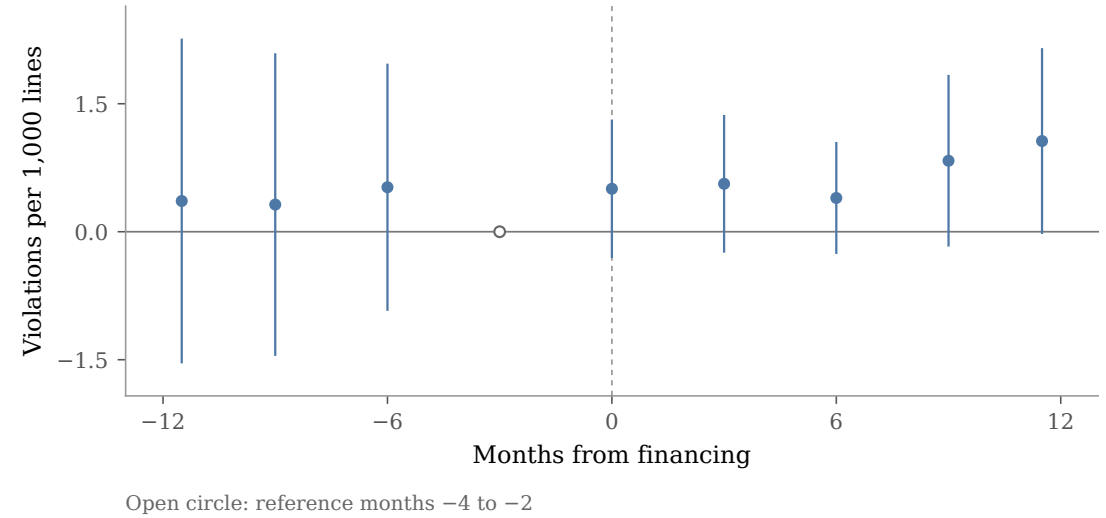


Figure 24: Later Financing: Critical Rule Violations per 1,000 Lines of Code. Controls remain without first institutional financing through the event window. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

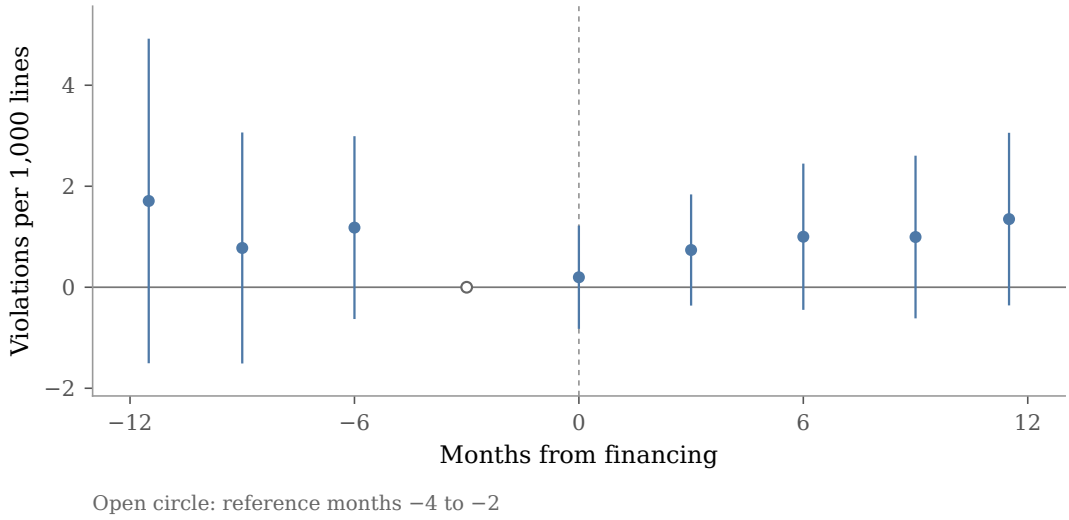


Figure 25: Later Financing: Rule Violations (All Severities) per 1,000 Lines of Code. Controls remain without first institutional financing through the event window. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

Table 28: Series B+ Placebo Difference-in-Differences Estimates for Critical Violations and All Violations

	Critical Violations			**Violations**		
	(1)	(2)	(3)	(4)	(5)	(6)
Treated × Post	0.368	0.298*	0.509*	0.131	−0.261	−0.075
	(0.567)	(0.160)	(0.271)	(0.964)	(0.540)	(0.729)
Post × High Violations (=1)		−1.455***			−2.052*	
		(0.370)			(1.060)	
Treated × Post × High Violations (=1)		0.293			1.018	
		(1.174)			(1.875)	
Post × Large Round			−0.290			−1.377
			(0.462)			(0.851)
Treated × Post × Large Round			−0.101			1.517
			(1.289)			(1.919)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	331,893	330,508	331,893	331,893	330,508	331,893
R^2	0.870	0.872	0.870	0.869	0.872	0.869

This table reports Series B+ placebo estimates from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Series B+ Treated × Post indicator. *Critical Violations* count severe SonarQube rule violations per 1,000 non-comment lines of code, including risky code structures with a potentially substantial impact on reliability, security, or maintainability. *Violations (All)* denote total deviations from engineering best practices across all severity levels per 1,000 lines of code. See Appendix A, Table 19, for full variable definitions.

High Violations is defined from critical-violation density in the month before the placebo event. Large Round refers to the amount raised at the earlier Seed or Series A treatment round. The quality cutoff uses baseline firm–cohort observations of firms that appear as placebo-treated; the round-size cutoff uses one observation per placebo-treated firm.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

Table 29: Series B+ Placebo Difference-in-Differences Estimates for Maintainability Issues

	Maintainability Issues (All)			**Maintainability Issues (High)**			**Maintainability Issues (Low)**		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Treated × Post	0.256	−0.204	−0.010	0.375	0.305*	0.529*	−0.386	−0.197	−0.581
	(0.867)	(0.551)	(0.709)	(0.567)	(0.159)	(0.270)	(0.465)	(0.258)	(0.491)
Post × High Violations (=1)		−1.881*			−1.459***			0.282	
		(1.063)			(0.370)			(0.715)	
Treated × Post × High Violations (=1)		1.138			0.297			−0.418	
		(1.642)			(1.172)			(0.811)	
Post × Large Round			−1.480*			−0.294			−0.857*
			(0.847)			(0.462)			(0.457)
Treated × Post × Large Round			1.732			−0.127			1.096**
			(1.606)			(1.289)			(0.526)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	331,893	330,508	331,893	331,893	330,508	331,893	331,893	330,508	331,893
R^2	0.859	0.862	0.859	0.869	0.871	0.869	0.843	0.846	0.843

This table reports Series B+ placebo estimates from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Series B+ Treated × Post indicator. Maintainability Issues (All) capture total maintainability-related rule violations per 1,000 lines of code. High-Severity Maintainability Issues identify maintainability problems assigned high impact, while Low-Severity Maintainability Issues capture more minor structural or readability deficiencies. See Appendix A, Table 19, for full variable definitions.

High Violations is defined from critical-violation density in the month before the placebo event. Large Round refers to the amount raised at the earlier Seed or Series A treatment round. The quality cutoff uses baseline firm–cohort observations of firms that appear as placebo-treated; the round-size cutoff uses one observation per placebo-treated firm.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

Table 30: Series B+ Placebo Difference-in-Differences Estimates for Security Hotspots

	Security Hotspots		
	(1)	(2)	(3)
Treated × Post	0.074	0.038	0.085
	(0.085)	(0.127)	(0.095)
Post × High Violations (=1)		0.077	
		(0.108)	
Treated × Post × High Violations (=1)		0.066	
		(0.116)	
Post × Large Round			−0.109
			(0.106)
Treated × Post × Large Round			0.057
			(0.119)
Cohort × Firm FE	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes
# Obs	331,893	330,508	331,893
R^2	0.701	0.702	0.701

This table reports Series B+ placebo estimates from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Series B+ Treated × Post indicator. *Security Hotspots* measure security-sensitive code segments requiring manual audit per 1,000 lines of code. See Appendix A, Table 19, for full variable definitions.

High Violations is defined from critical-violation density in the month before the placebo event. Large Round refers to the amount raised at the earlier Seed or Series A treatment round. The quality cutoff uses baseline firm–cohort observations of firms that appear as placebo-treated; the round-size cutoff uses one observation per placebo-treated firm.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

Table 31: Series B+ Placebo Difference-in-Differences Estimates for Bugs and Reliability Issues

	Bugs			**Reliability Issues (High)**			**Reliability Issues (Low)**		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Treated × Post	−0.132	−0.046	−0.061	−0.011	−0.008	−0.022	−0.148	−0.202	−0.148
	(0.164)	(0.060)	(0.103)	(0.010)	(0.012)	(0.014)	(0.126)	(0.189)	(0.127)
Post × High Violations (=1)		−0.194**			0.005			−0.022	
		(0.093)			(0.017)			(0.137)	
Treated × Post × High Violations (=1)		−0.156			−0.005			0.114	
		(0.333)			(0.021)			(0.222)	
Post × Large Round			0.085			0.002			−0.040
			(0.066)			(0.008)			(0.103)
Treated × Post × Large Round			−0.225			0.024			0.032
			(0.376)			(0.016)			(0.212)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	331,893	330,508	331,893	331,893	330,508	331,893	331,893	330,508	331,893
R^2	0.907	0.907	0.907	0.501	0.501	0.501	0.891	0.891	0.891

This table reports Series B+ placebo estimates from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Series B+ Treated × Post indicator. *Bugs* capture code patterns classified by SonarQube as likely to cause incorrect or unexpected behavior, such as invalid operations or failures to handle an input per 1,000 lines of code. *High-Severity Reliability Issues* represent defects posing significant risk to system stability, while *Low-Severity Reliability Issues* denote minor or edge-case flaws with limited immediate impact. See Appendix A, Table 19, for full variable definitions.

High Violations is defined from critical-violation density in the month before the placebo event. Large Round refers to the amount raised at the earlier Seed or Series A treatment round. The quality cutoff uses baseline firm–cohort observations of firms that appear as placebo-treated; the round-size cutoff uses one observation per placebo-treated firm.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

Table 32: Series B+ Placebo Difference-in-Differences Estimates for Technical Debt

	Technical Debt			**Technical Debt Ratio**		
	(1)	(2)	(3)	(4)	(5)	(6)
Treated × Post	−0.128	−0.151	−0.315	−0.631	−0.557	−1.165
	(0.407)	(0.205)	(0.396)	(1.563)	(0.767)	(1.504)
Post × High Violations (=1)		−0.368			−1.286	
		(0.616)			(2.346)	
Treated × Post × High Violations (=1)		0.087			−0.023	
		(0.757)			(2.849)	
Post × Large Round			−0.556			−1.925
			(0.358)			(1.314)
Treated × Post × Large Round			0.850			2.687
			(0.621)			(2.299)
Cohort × Firm FE	Yes	Yes	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes	Yes	Yes
# Obs	331,893	330,508	331,893	320,132	318,756	320,132
R^2	0.840	0.841	0.840	0.840	0.841	0.840

This table reports Series B+ placebo estimates from the specification $M_{i,c,t} = \beta VC_{i,c,t} + \chi_{c,i} + \delta_{c,t} + \gamma X_{i,c,t} + \varepsilon_{i,c,t}$, where $VC_{i,c,t}$ is the Series B+ Treated × Post indicator. *Technical Debt per Line of Code* measures calibrated remediation costs in dollars per line of code, while the *Technical Debt Ratio* expresses that same calibrated debt as a percentage of the firm's software assets. See Appendix A, Table 19, for full variable definitions.

High Violations is defined from critical-violation density in the month before the placebo event. Large Round refers to the amount raised at the earlier Seed or Series A treatment round. The quality cutoff uses baseline firm–cohort observations of firms that appear as placebo-treated; the round-size cutoff uses one observation per placebo-treated firm.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

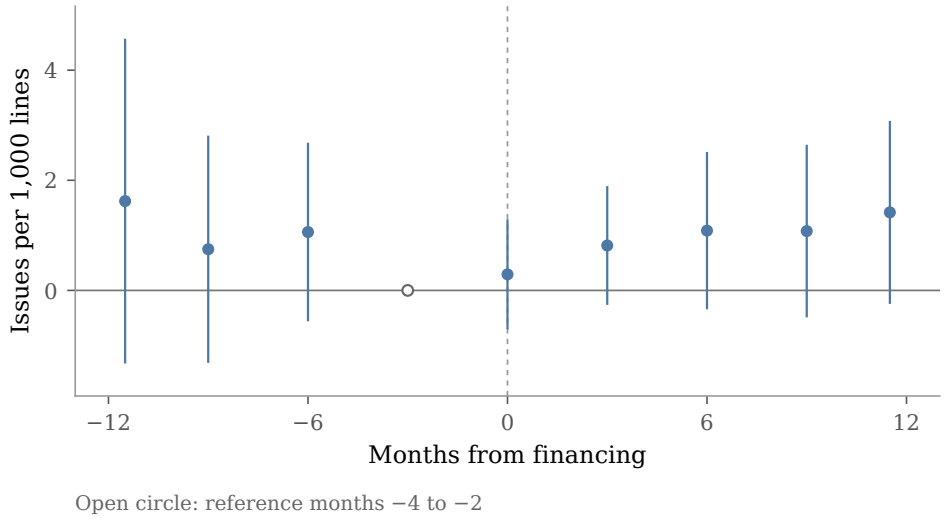


Figure 26: Later Financing: Maintainability Issues per 1,000 Lines of Code. Controls remain without first institutional financing through the event window. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

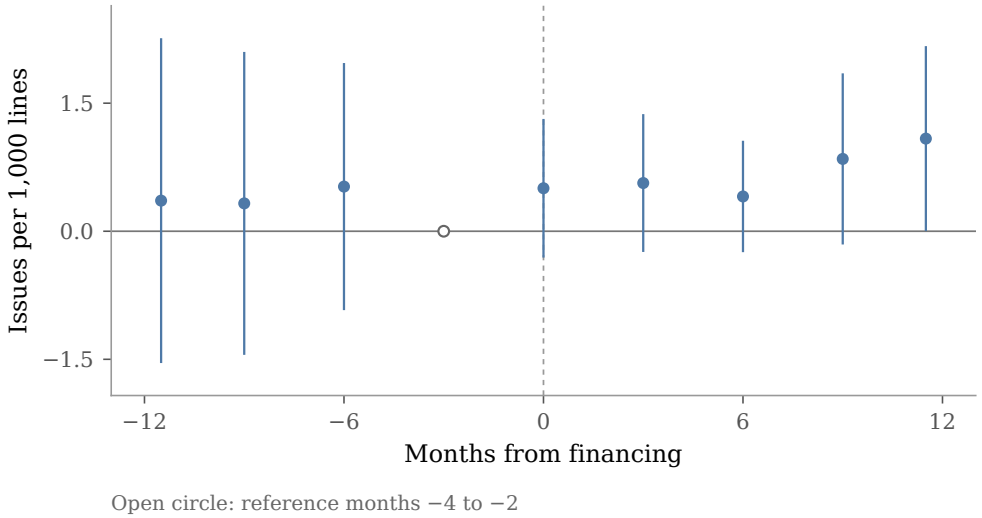


Figure 27: Later Financing: Maintainability Issues (High-Severity) per 1,000 Lines of Code. Controls remain without first institutional financing through the event window. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

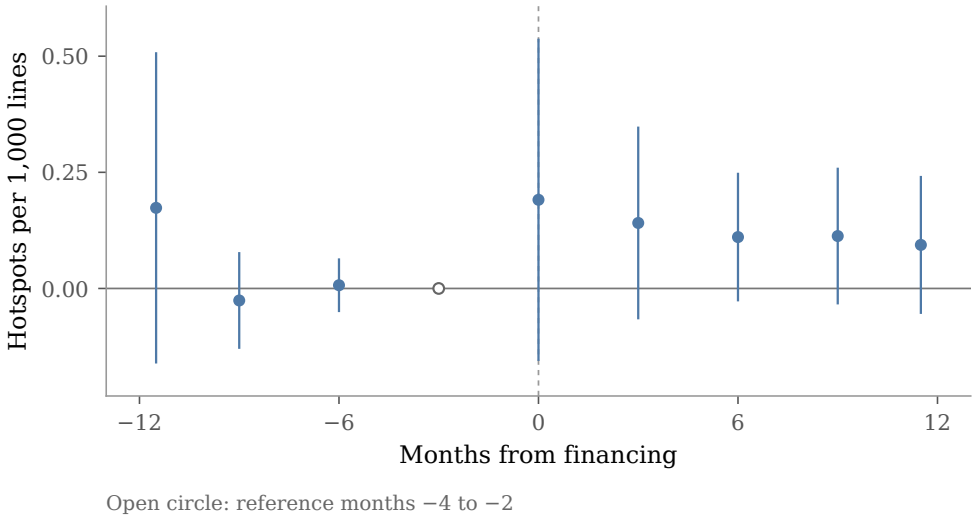
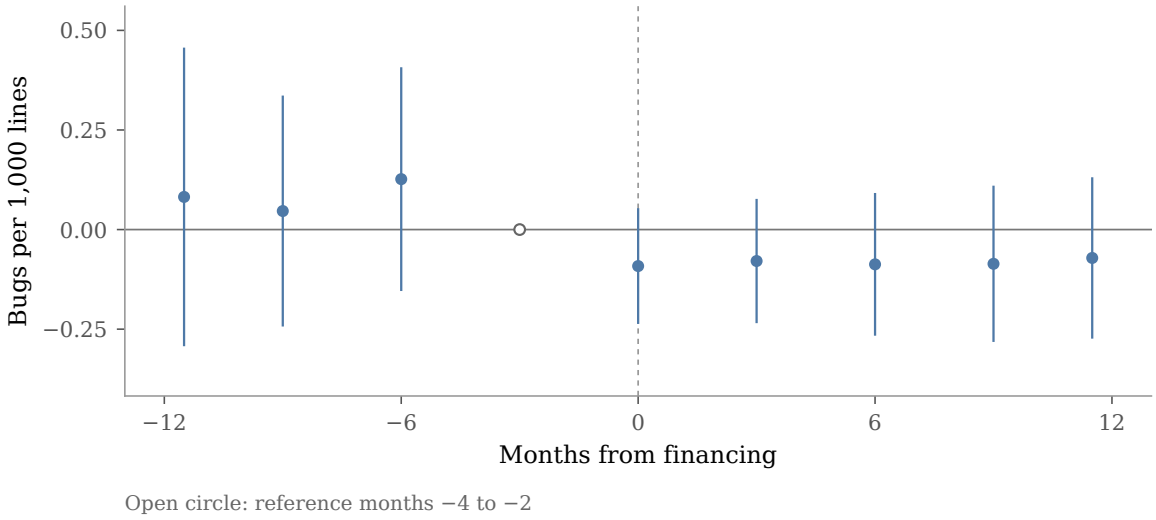
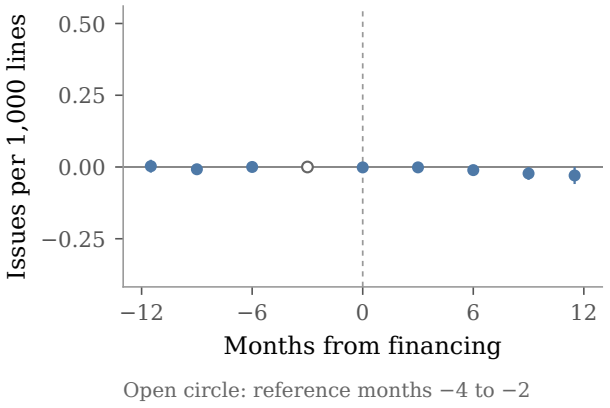


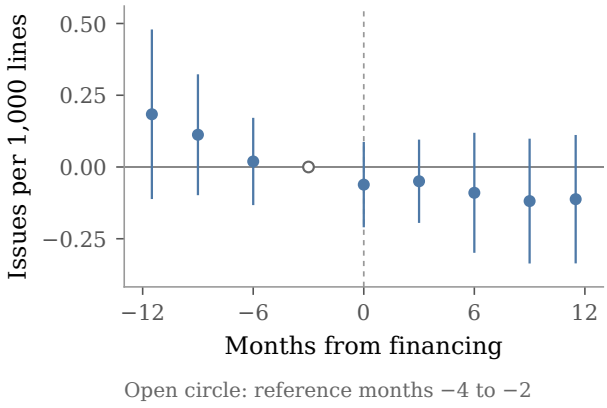
Figure 28: Later Financing: Security Hotspots per 1,000 Lines of Code. Controls remain without first institutional financing through the event window. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.



(a) Bugs per 1,000 lines of code

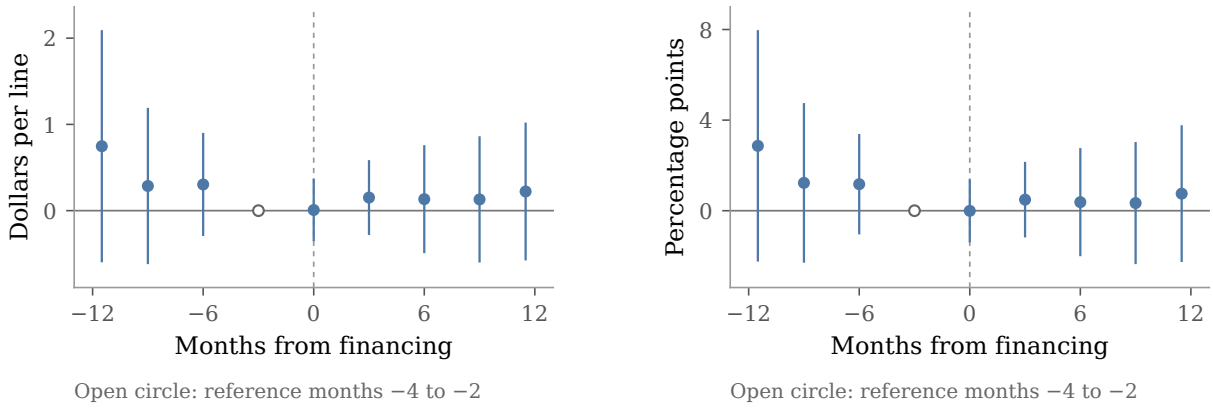


(b) Reliability Issues (High Severity) per 1,000 lines of code



(c) Reliability Issues (Low Severity) per 1,000 lines of code

Figure 29: Later Financing: Bugs and Reliability Issues. Controls remain without first institutional financing through the event window. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.



(a) Technical Debt per Line of Code ($)

(b) Technical Debt Ratio

Figure 30: Later Financing: Technical Debt Measures. Controls remain without first institutional financing through the event window. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

K Lifecycle and Commercialization: Additional Evidence

This appendix examines whether the changes around first institutional financing coincide with a shift toward commercialization. It complements the commercial-stage comparisons in Section 4.3 with LinkedIn employment records and commit histories. We use the earlier single-repository panel, which contains 212 treated firms and 177,676 stacked observations for the commit measures, and the estimator in Equation (7). These analyses cover a subset of the main sample; each table reports its own coverage.

K.1 Headcount Composition

We use employment records to test whether commercialization shifts resources from engineering toward selling. We estimate Equation (7) on each role group's share of active headcount. Table 33 reports a 2.35 percentage-point increase in the business-development share and a 6.65-point increase in the engineering share, both significant at the 1% level. Business-development roles also account for 4.23 percentage points more of monthly hires. Firms therefore expand commercial capacity alongside a larger increase in engineering's share of employment, as the event studies in Figure 31 also illustrate.

Table 33: Business-Development and Engineering Shares of Headcount

	Business Development Share of Headcount	Engineering Share of Headcount	Business Development Share of Monthly Hires
	(1)	(2)	(3)
Treated × Post	0.0235***	0.0665***	0.0423***
	(0.0068)	(0.0149)	(0.0119)
Cohort × Firm FE	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes
# Obs	157,553	157,553	68,416
# Firms	287	287	279

This table estimates the baseline stacked difference-in-differences specification of Equation (7) on LinkedIn employment spells for the single-repository panel firms with LinkedIn coverage. The outcomes in Columns (1) and (2) are the business-development and software-engineering categories' shares of a firm's active headcount in a month; the two categories are mutually exclusive. Column (3) uses the flow of role starts rather than the stock of active employees. Joint tests do not reject zero pre-treatment coefficients for these three share outcomes. They reject for the corresponding log-count outcomes, which are not reported. These outcomes derive from employment records and contain no SonarQube input.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

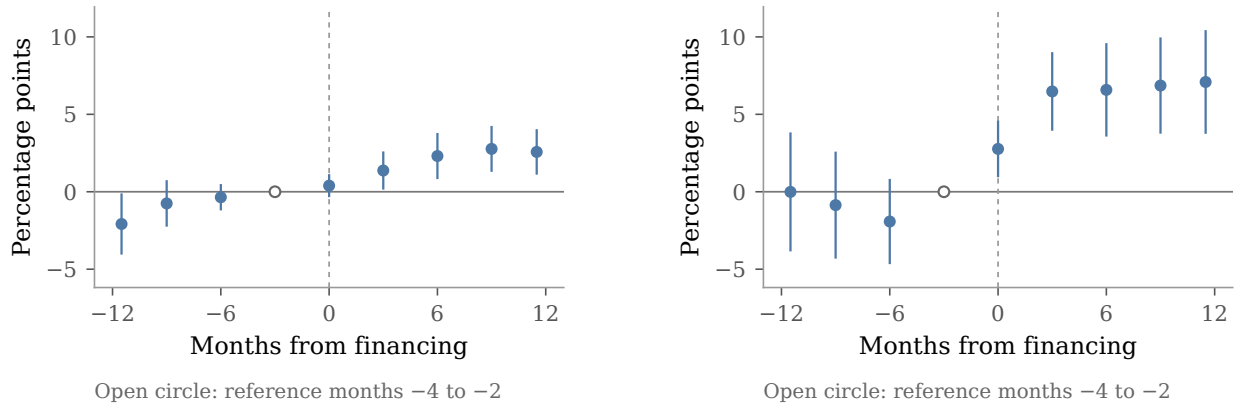
Table 34 decomposes the commercial expansion into sales-related and marketing roles and identifies human-resources staff within the residual category. Sales-related roles account for 1.50 percentage points of the increase in headcount share ($p < 0.05$), and marketing for 0.87 points ($p < 0.01$). The human-resources estimate is statistically insignificant. The share of all other classified roles falls by 7.62 points ($p < 0.01$), and the reported hiring-flow estimates have the same signs. These changes indicate joint investment in engineering and commercial capacity, with no transfer of employment share from engineering toward selling.

Table 34: Decomposition of the Business-Development Share of Headcount and Hires

	Sales-Related	**Marketing**	**Human Resources**	**All Other Roles**
	(1)	(2)	(3)	(4)
Panel A: Share of active headcount				
Treated × Post	0.0150**	0.0087***	−0.0021	−0.0762***
	(0.0064)	(0.0031)	(0.0060)	(0.0155)
# Obs	157,553	157,553	157,553	157,553
# Firms	287	287	287	287
Panel B: Share of monthly hires				
Treated × Post	0.0234**	0.0189***	—	−0.1059***
	(0.0114)	(0.0060)		(0.0252)
# Obs	68,416	68,416	—	68,416
# Firms	279	279	—	279
Cohort × Firm FE	Yes	Yes	Yes	Yes
Month × Cohort FE	Yes	Yes	Yes	Yes

This table decomposes the business-development share from Table 33, using the same stacked difference-in-differences specification, sample, and employment records. Columns (1) and (2) split the business-development bucket into sales-related roles (sales, account management, sales development, business-development titles, partnerships, customer success, go-to-market, and revenue operations) and marketing roles (marketing, product marketing, growth, and demand generation); the two coefficients sum to the aggregate business-development estimate up to a small overlap from employees who hold roles in both components. Column (3) is a human-resources bucket (human resources, recruiting, talent acquisition, and people operations) drawn from the previously unclassified residual, and Column (4) is the remaining residual. Engineering-management roles are excluded from every bucket, following the classifier of Table 33, so the reported shares and the engineering share do not sum to one. Panel A uses each bucket's share of a firm's active headcount; Panel B uses the share of monthly role starts. Joint tests do not reject zero pre-treatment coefficients for the reported outcomes. They reject for the human-resources share of monthly hires, which is not reported. These outcomes derive from employment records and contain no SonarQube input.

Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.



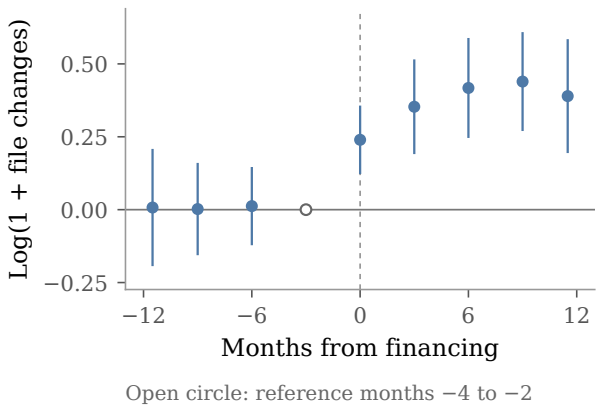
(a) Business-development share of headcount

(b) Engineering share of headcount

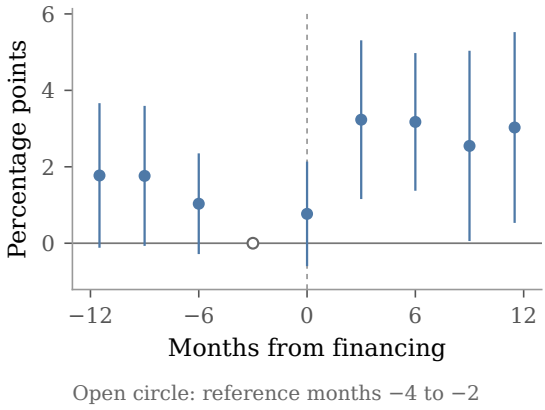
Figure 31: Headcount Composition. Single-repository sample; shares in percentage points. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

K.2 Commercialization in the Codebase

We also examine commercialization within the software asset using commit messages, changed file paths, and dependency manifests—the files that list the external software packages used by a project. Table 18 shows no statistically significant increase in messages mentioning billing, payments, metering, subscriptions, or upgrades, or in the share of changed files associated with commercial features, payment libraries, enterprise functions, or growth. In the year after financing, 1.0% of repositories add a payment library, while 7.3% add a frontend framework used to build the user-facing interface. Development activity nevertheless expands: file changes per month rise by 34 log points, engineers touch more top-level directories, and the share of changes to frontend code increases. Figure 32 traces these changes over the event window. The observable expansion is therefore broader than a shift toward the measured monetization features.



(a) Log file changes per month



(b) Frontend share of changed files

Figure 32: Development Activity and the Frontend Share of Changed Files. Single-repository sample; shares in percentage points. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

Together, the employment and codebase evidence does not support a narrow lifecycle account in which firms shift resources from engineering toward selling at their first round. Engineering gains employment share, development expands, and the public codebase shows no detectable increase in the measured monetization features. Combined with the quality improvements among firms already generating revenue in Section 4.3, these findings indicate that the engineering response extends beyond the initial transition to commercialization.

L Treatment Round Type: Seed versus Series A

This appendix tests whether engineering improvements differ between firms entering institutional financing at Seed and those entering at Series A. If the main result reflects a particular stage of firm development, we might expect different quality responses across these financing rounds. We estimate a pooled triple-difference with Seed-treated firms as the base group, accounting for the covariance induced by the two groups' shared controls. Table 35 reports negative estimates for both groups on every outcome. None of the differences is statistically significant: +0.66 critical violations per 1,000 lines ($p = 0.72$), +2.53 total violations per 1,000 lines ($p = 0.45$), \$1.63 of technical debt per line, and +5.94 percentage points of the Technical Debt Ratio ($p = 0.17$ for both debt measures).

The pooled model does not reject equal pre-treatment paths by round type (joint p-values between 0.45 and 0.55). Figure 33 shows the corresponding quarterly estimates. The critical-violation difference remains insignificant with never-treated controls only (+0.79, $p = 0.67$) and after adding a Series A × Post term (+0.08, $p = 0.97$). The results provide no evidence that the quality gains differ systematically between Seed and Series A entry.

The pooled comparison is the more reliable source of evidence on stage differences: the separate Series A subsample rejects its own joint pre-trend test ($p = 0.006$), despite the absence of differential leads in the pooled model. The 72 Series A firms are also older at entry and raise roughly three times as much capital as the Seed firms. Round type therefore combines several dimensions of maturity and financing. The direct commercial-status comparison in Section 4.3 addresses the more specific hypothesis that quality improves when a firm first begins generating revenue.

Table 35: Treatment Round Type: Seed versus Series A

	Seed Estimate	Series A Estimate	Difference (Series A − Seed)
Critical Violations / kLoC	−2.016**	−1.359	0.656
	(1.012)	(1.577)	(1.851)
Violations (All) / kLoC	−5.384**	−2.859	2.526
	(2.563)	(2.228)	(3.336)
Technical Debt per LoC ($)	−1.980**	−0.346	1.634
	(0.972)	(0.734)	(1.169)
Technical Debt Ratio	−7.200**	−1.258	5.942
	(3.535)	(2.669)	(4.250)
Cohort × Firm FE		Yes	
Month × Cohort FE		Yes	
# Obs		487,577	

Pooled stacked difference-in-differences on the full estimation sample (202 Seed-treated and 72 Series A-treated firms), with Seed-treated firms as the base group. The Series A column reports the linear combination of the base effect and the interaction; the Difference column reports the interaction coefficient itself, whose t-test is the test of equality of the two round-type effects. Critical and total violations are per 1,000 lines of code; technical debt is dollars per line; the Technical Debt Ratio is calibrated technical debt as a percentage of software assets. Joint tests show no evidence of differential pre-treatment leads by round type for any of the four outcomes. See Appendix A, Table 19, for variable definitions.

Note: Standard errors in parentheses, clustered by cohort and firm. * p<0.1, ** p<0.05, *** p<0.01.

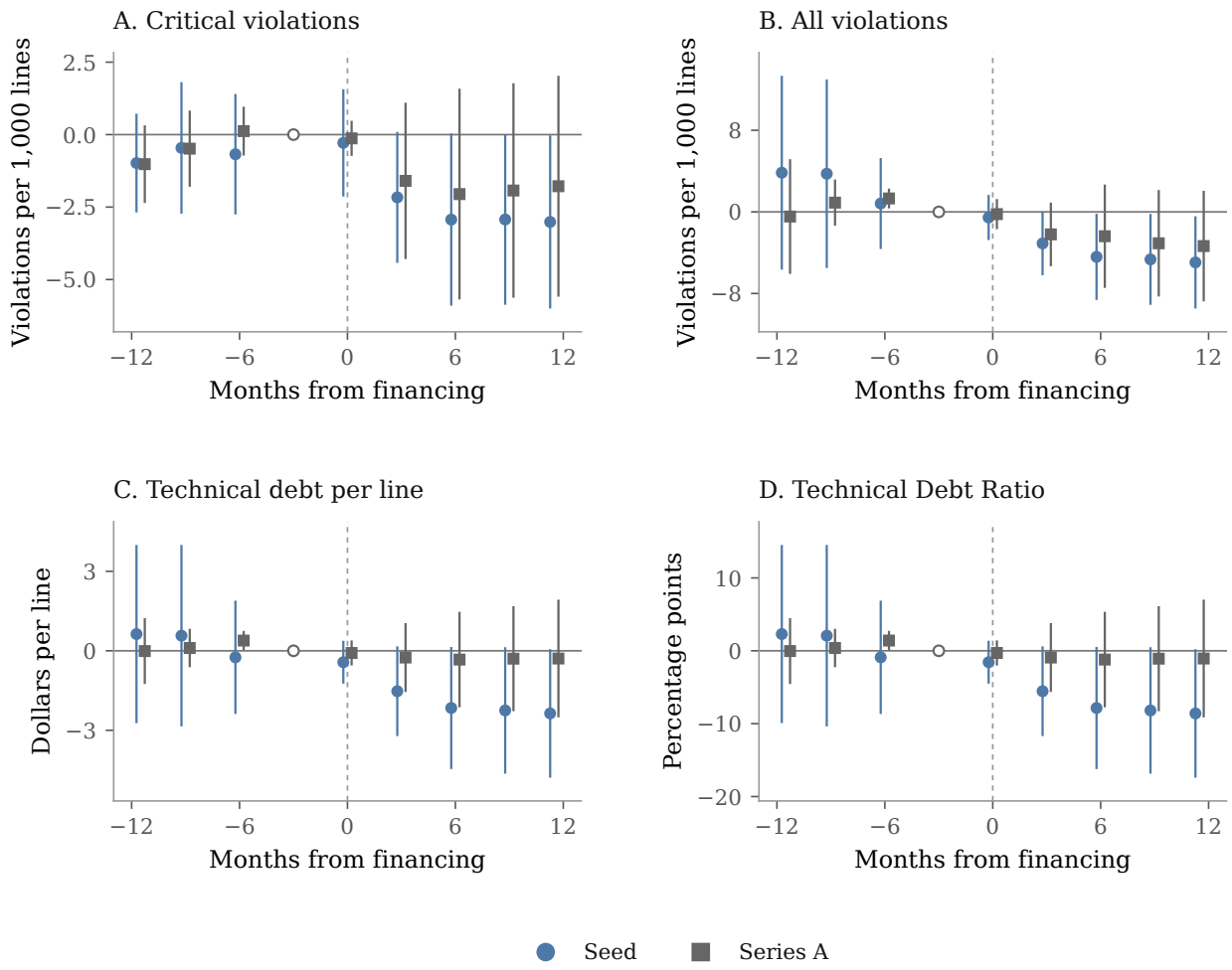


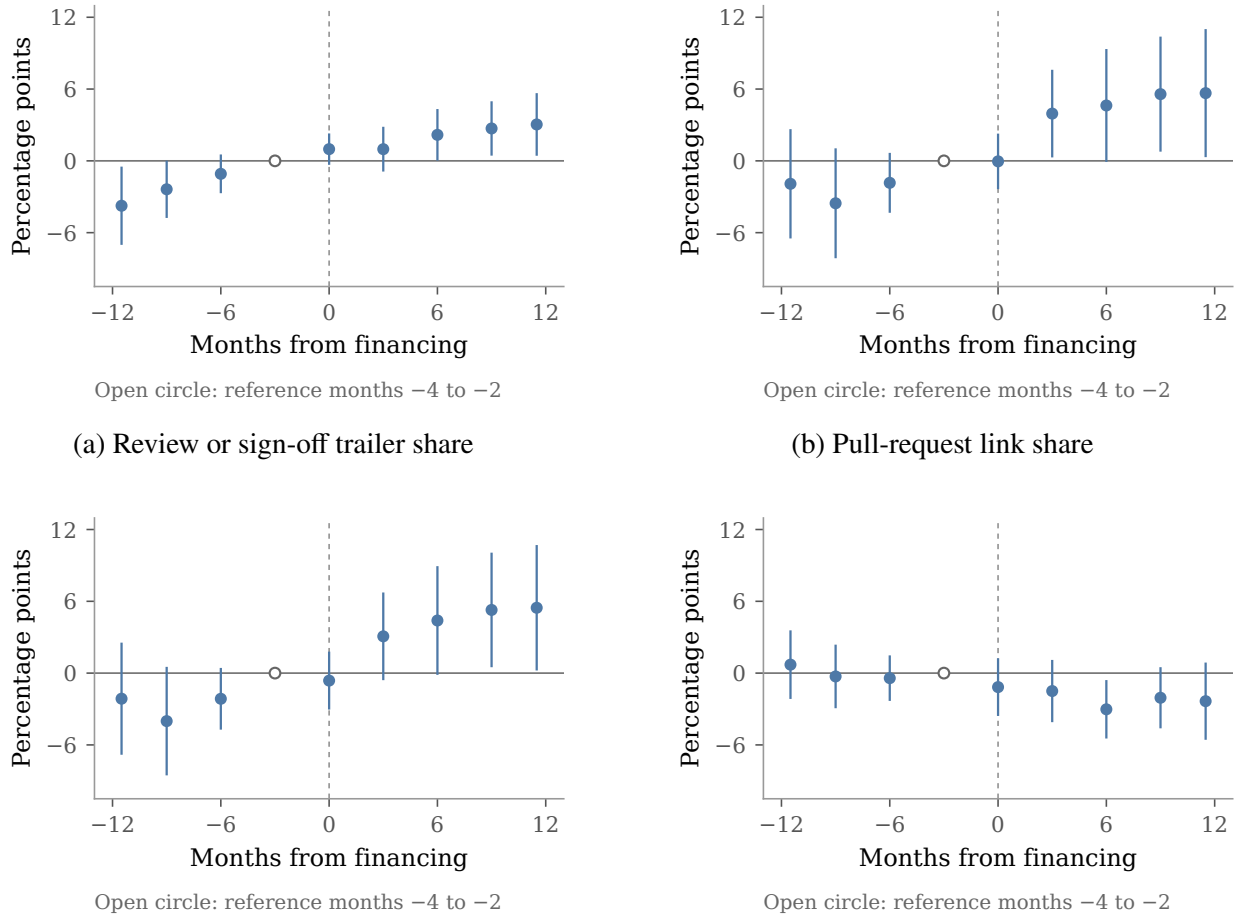
Figure 33: Engineering Quality Around Seed and Series A Entry. Paths from the pooled model in Table 35; Series A intervals account for covariance between the base and interaction estimates. The Technical Debt Ratio is in percentage points. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

M Engineering Practice around VC Entry

This appendix examines the changes in development practices behind the evidence in Section 3.8. We report process-measure event studies, later-round comparisons, commit-timing estimates, and defect attribution using single-repository commit histories. The commit-message measures cover 212 treated firms and 177,676 stacked observations; the defect-attribution panel covers 273 firms and 211,511 stacked observations. Firm–month outcomes use the specification and clustering in Equation (7), while the commit-level timing regressions use the fixed effects described below.

M.1 Engineering Process Measures

We construct process measures from 4.8 million commit messages to assess how development procedures change after financing. Table 11 reports the share of commits carrying review, pull-request, or issue-tracking markers and the share authored outside business hours. Figure 34 shows the event studies for four measures, with joint tests that do not reject zero pre-treatment coefficients. The rise in documented review and coordination, together with the shift toward regular hours, places the changes in development practice around first institutional entry.



(a) Review or sign-off trailer share

(b) Pull-request link share

(c) Issue-reference share

(d) Off-hours commit share

Figure 34: Engineering Process Measures. Single-repository sample; shares in percentage points. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

The later-round comparison produces a different pattern. Around Series B and later rounds, the issue-reference share falls by 4.6 percentage points and the merge-or-squash share by 4.7 percentage points, both statistically significant. The latter compares with a 6.0-point increase in merge-or-squash activity at first entry. The integration marker therefore does not show recurring gains with each

financing round, consistent with an initial change in how development is organized under institutional backing.

M.2 Commit Timing and Defect Risk

We next ask whether commit timing is associated with subsequent defect attribution. For non-merge, non-bot commits written through mid-2024, we relate an indicator for attribution to a defect fixed within 24 months to off-hours, weekend, and late-night indicators, with author-by-repository and calendar-month fixed effects. Table 36 reports an off-hours coefficient of 0.86 percentage points, about 5% of the 17.7% base rate in the commit-level sample, and a weekend coefficient of 0.61 points; both are significant at the 1% level. The late-night indicator is nested within off-hours, so its coefficient is an additional contrast within that window. The associations compare contributions by the same engineer in the same repository, while allowing that timing can correlate with task difficulty, deadlines, and commit size.

Table 36: Commit Timing and Defect Introduction

Commit timing	Pr(introduces a defect fixed within 24 months)
Off-hours (20:00–05:59)	0.0086***
	(0.0012)
Weekend	0.0061***
	(0.0013)
Late-night increment (00:00–05:59)	−0.0049***
	(0.0014)
Author × Repository FE	Yes
Calendar-month FE	Yes
# Obs	3,156,028

Linear probability estimates on 3,156,028 non-merge, non-bot commits written through June 2024 across 268 covered single-repository firms and 104,141 authors. The dependent variable equals one if the attribution procedure links the commit to a bug fix within 24 months; the pre-treatment base rate in this commit-level sample is 17.7%. The regression includes the three timing indicators jointly and controls for message-length deciles. Timing is measured in UTC. Off-hours is 20:00–05:59; late-night is the 00:00–05:59 subset, so the late-night coefficient is additional to the off-hours coefficient. Replacing the off-hours measure with one based on each author's modal local hours raises its coefficient to 0.0093*** (0.0016), while the late-night coefficient becomes −0.0008 ($p = 0.61$). With repository fixed effects but without author fixed effects, the off-hours coefficient is 0.0137***. For commits linked to security-related fixes, it is 0.0006*** on a 0.75% base. Commit size beyond message length is unobserved.

Note: Standard errors, clustered by firm, in parentheses. * p<0.1, ** p<0.05, *** p<0.01.

M.3 Defect Attribution to Incumbent Engineers

To examine whether quality improvements reach the existing team, we trace bug-fix commits to earlier changes using `git blame`, which identifies the last commit to modify each repaired line. We classify its author as an incumbent if the engineer was active before financing and as a newcomer otherwise. This procedure links later repairs to candidate defect-introducing contributions, allowing us to examine changes in the output of developers already present at the firm.

Table 37 shows that the quality improvement extends to incumbent engineers. Their defect-attribution rate falls by 1.76 percentage points after the first round ($p = 0.048$), roughly 13% of the 13.5% pre-financing rate in the firm–month attribution panel. Figure 35 shows that the decline follows financing, with no evidence of differential pre-treatment leads. The newcomer interaction is statistically insignificant, and the firm-wide rate declines by 1.15 percentage points ($p = 0.162$). Newcomers account for a larger share of attributed defects after financing, reflecting their entry into the development team; because that share is zero before entry by construction, it describes the composition of post-financing contributions.

Table 37: Defect Attribution: Incumbents Versus New Hires

Comparison	Dependent Variable	Estimate	p-value
Incumbents	Incumbent defect-attribution rate	−0.0176**	0.048
		(0.0088)	
Newcomer differential	Treated × Post × Newcomer	0.0054	0.387
		(0.0062)	
Post-entry composition	Newcomer share of introducing commits	0.1362***	<0.001
		(0.0273)	
Firm-wide	Total defect-attribution rate	−0.0115	0.162
		(0.0081)	
Cohort × Firm FE	Yes		
Month × Cohort FE	Yes		
# Firms (incumbent regression)	273		

This table estimates Equation (7) on defect-attribution outcomes from the single-repository panel. A group's defect-attribution rate is the fraction of its commits linked to a subsequent bug fix. The newcomer share is the fraction of defect-introducing commits attributed to engineers who joined after financing. The newcomer interaction compares group rates and the total rate pools all engineers. The incumbent regression uses 211,511 observations across 273 firms; the other three specifications use 284, 283, and 284 firms, respectively. The incumbent joint pre-trend test returns $p = 0.99$. Its coefficient is −0.0173 ($p = 0.051$) after dropping the five largest repositories and −0.0183 ($p = 0.064$) in the subsample with high attribution coverage. Newcomer shares are zero before entry by construction, so their post-entry increase does not have the same difference-in-differences interpretation as the incumbent rate. Attribution depends on the observation of later fixes, and the incumbent rate compares the contributions of a group whose active membership can change over time.
Note: Standard errors in parentheses. * p<0.1, ** p<0.05, *** p<0.01. Standard errors are clustered by cohort and firm.

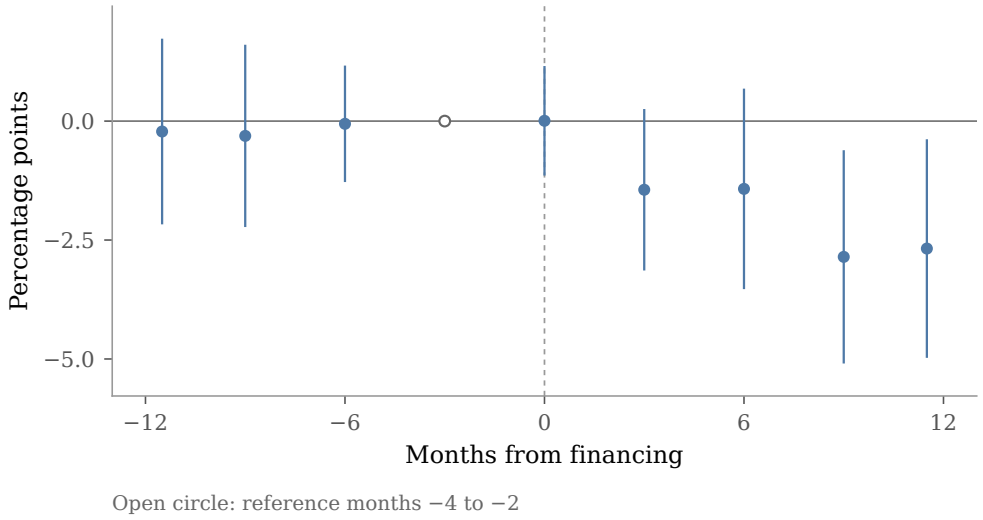


Figure 35: Incumbent Defect-Attribution Rate. Single-repository sample; shares in percentage points. Quarterly estimates are relative to months −4 to −2; the zero bin spans months −1 to +1. Bars show 95% confidence intervals; standard errors are clustered by firm and cohort.

The process markers and incumbent estimates jointly support the professionalization interpretation. Increased documentation of review and tracking accompanies a lower fraction of incumbent contributions linked to later fixes, showing that the changes reach the existing team. However, combining the timing coefficients with the changes in off-hours shares accounts for approximately 1.6% of the incumbent decline, using overlapping samples. The shift in working hours is therefore one marker of a broader organizational change, alongside more formal review and coordination. The contribution histories provide evidence that engineering practices improve within firms after first institutional entry.

N Additional Outcomes and Design Sensitivity

This appendix reports sensitivity to the event window, treatment timing, and control reuse, followed by additional outcomes and evidence on process adoption, repository allocation, and incumbent retention. The outcome and repository-allocation analyses use the main sample. The process-adoption analysis uses the 394 firms covered by the commit corpus.

N.1 Design and Inference Sensitivity

Table 38 collects the principal variations of the baseline design. The tests address three distinct concerns: sensitivity to observations immediately around the deal, the period over which firms are compared, and the weight placed on repeatedly used controls. Estimates remain negative under

donut exclusions, alternative windows, and inverse-frequency weighting. The persistence across these choices indicates that the quality improvement is not confined to the financing month or to one construction of the comparison sample.

Table 38: Additional Design Sensitivity

Specification	Critical Violations	Technical Debt	Violations
Baseline (±12 months)	−1.849**	−1.566**	−4.744**
	(0.863)	(0.765)	(2.016)
Donut radius 1	−1.970**	−1.731**	−5.204**
	(0.943)	(0.846)	(2.225)
Donut radius 2	−2.065**	−1.840**	−5.616**
	(0.986)	(0.908)	(2.426)
±6 months	−1.714**	−0.919*	−2.473**
	(0.778)	(0.484)	(1.033)
±9 months	−1.847**	−1.331**	−3.880**
	(0.803)	(0.606)	(1.503)
±18 months	−2.248**	−2.230**	−6.139**
	(0.993)	(1.005)	(2.747)
±24 months	−2.297*	−1.705**	−4.328**
	(1.155)	(0.810)	(1.766)
Inverse control-frequency weights	−1.976**	−1.830**	−5.295**
	(0.892)	(0.917)	(2.523)

Critical Violations and Violations are counts per 1,000 lines of code; Technical Debt is calibrated dollars per line. Donut specifications omit observations within the stated number of months on both sides of treatment. Inverse-frequency weights down-weight controls in proportion to the number of cohort stacks in which they appear. Standard errors are clustered by cohort and firm. * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

Across 99 leave-one-cohort-out estimates, the coefficient ranges are $[-2.00, -1.25]$ for Critical Violations, $[-1.67, -1.06]$ for Technical Debt, and $[-5.00, -3.15]$ for Violations; every estimate is negative. In 500 within-cohort permutations, permutation p-values are < 0.002, 0.028, and 0.002, respectively. These permutation comparisons rely on exchangeability of treatment labels within a cohort; financing was not randomly assigned. The corresponding two-way-clustered asymptotic p-values are 0.035, 0.043, and 0.021.

N.2 Additional SonarQube Outcomes

We examine 42 additional SonarQube outcomes grouped into nine families, with expected signs, pre-trend tests, and selection criteria specified before estimating this supplementary battery. The analysis applies the baseline specification, a Series B+ placebo, a five-repository cap, a single-repository restriction, and count-model checks where applicable. Table 39 reports selected results

and multiplicity-adjusted significance levels, allowing us to assess whether the main findings extend to other dimensions of the codebase.

Table 39: Additional SonarQube Outcomes

Outcome	First VC Entry	Series B+ Placebo
Log duplicated lines	0.358***	0.014
	(0.099)	(0.061)
Duplicated blocks per 1,000 LoC	0.942**	0.194
	(0.448)	(0.309)
Code smells per 1,000 LoC	−4.803**	0.104
	(1.990)	(0.889)
Complexity per 1,000 LoC	−17.371*	9.338
	(9.938)	(5.725)

Each cell is a Treated × Post coefficient. Log duplicated lines is ln(1+duplicated lines). Duplicated blocks counts repeated code blocks; complexity counts linearly independent paths through the code. Both are divided by non-comment lines of code and multiplied by 1,000. Code smells are SonarQube's legacy maintainability taxonomy and provide a classification robustness check rather than a distinct economic outcome. Log duplicated lines has a Benjamini–Hochberg q-value of 0.001; duplicated blocks has $q = 0.114$. Complexity is suggestive at the 10% level. The screen also finds null density estimates throughout the security family and nonmonotone estimates across the full severity ladder. Standard errors are clustered by cohort and firm. * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$.

The duplication results identify a distinct margin on which engineering quality does not improve. Duplicated lines grow faster than code volume even as maintainability violations per line decline. Reusing code can accelerate development while creating a future maintenance burden, so these measures need not move together. The main result is therefore a reduction in rule violations and maintainability issues, accompanied by greater use of repeated code structures as firms expand.

N.3 Timing of Process Adoption

We define sustained adoption as a process marker appearing in at least 25% of commits for three consecutive observable months, requiring at least five commits in each month. A discrete-time hazard model includes duration and calendar-year fixed effects and clusters by firm. Among 394 firms covered by the commit corpus, the monthly hazard of sustained testing or continuous-integration adoption—automated building and testing of each code update—rises by 0.224 percentage points ($p = 0.046$) from a 0.47% base, a relative increase of approximately 48%. Similarly, a fixed-effects logit produces an odds ratio of 1.38 (95% CI $[1.02, 1.88]$). By contrast, issue references, review trailers, and standardized commit-message prefixes show no detectable timing shift.

The testing-adoption result is strongest at lower adoption thresholds and disappears at the 50% threshold, where few firms qualify as adopters. Adjustment across the four markers, none designated as a primary outcome, also removes conventional statistical significance. We therefore treat the adoption estimate as supplementary support for the more formal development process documented in the review and tracking measures.

N.4 Repository Allocation and Incumbent Retention

The repository-allocation test asks whether the quality improvement reflects concentrating development in a core repository while pruning peripheral projects. We designate the largest repository before financing as the flagship and track its subsequent share of activity. The multi-repository panel contains 55 treated firms; commit-based outcomes are available for 42 of them. The flagship's commit share changes by +0.031 ($p = 0.321$), and within-firm commit concentration by −0.006 ($p = 0.633$), with neither estimate statistically significant. Active repositories increase by 0.183 ($p = 0.061$), while new-repository creation is statistically insignificant. These results provide no evidence of a systematic shift toward the flagship or withdrawal from peripheral repositories.

Finally, we test whether the improvement among incumbents reflects the selective retention of engineers who already produced cleaner code. Fixing the engineer cohort at $t = -1$ and controlling for pre-period activity, we estimate a −2.8 percentage-point change in 12-month retention (95% CI $[-8.1, +2.4]$). A direct comparison of retained engineers with leavers yields a treated-firm differential of −0.17 percentage points (SE = 1.52, $p = 0.913$) in pre-period defect propensity. Neither test detects the selective-retention pattern that would explain the incumbent result, supporting an interpretation based on changes in engineering practices, although the confidence intervals permit economically meaningful selection.