The ROI of AI-assisted Software Development

Google Cloud

DORA

Contents

Unless otherwise noted, all citations retrieved February, 2026.

Executive summary

AI is an amplifier

Artificial intelligence (AI) serves as a powerful amplifier in software development. It magnifies the strengths of high-performing organizations and the dysfunctions of struggling ones. The greatest returns on AI investment come not from the tools themselves but from a strategic focus on the underlying organizational system: the quality of the internal platform, the clarity of workflows, and the alignment of teams. Without this foundation, AI creates localized pockets of productivity that are often lost in downstream chaos.

Because AI acts as an amplifier, it dramatically increases the volume of code generated. However, it is important to remember that code is often seen as a liability, not an asset.[1] Operational costs of running applications and services far outweigh the costs of building them. Generating more code without proper oversight can increase verification overhead and lead to long-term technical debt.

Quantify the return on investment (ROI) of AI investments

AI can impact many aspects of value throughout the software development lifecycle (SDLC): cost efficiency, security, productivity, developer experience, and user experience all contribute to business success. However, it is difficult to measure the direct impact of AI without an understanding of "before." Do you have a baseline? Establishing this foundation is critical, though maybe what matters more is that you're measuring it at all.

Navigate the J-Curve of AI value realization

Realism is essential when forecasting the timeline to ROI for AI. While AI acts as an amplifier, the path to value is rarely a straight line. We anticipate that most organizations will encounter a J-Curve: a temporary productivity dip and period of instability associated with early adoption. Think of this dip as the tuition cost of transformation.

It isn't a signal of failure but a necessary investment in learning and adaptation. By explicitly forecasting this cost and investing in the underlying engineering systems, leaders can help their teams navigate this volatility, quickly pivoting from initial disruption to long-term value and innovation.

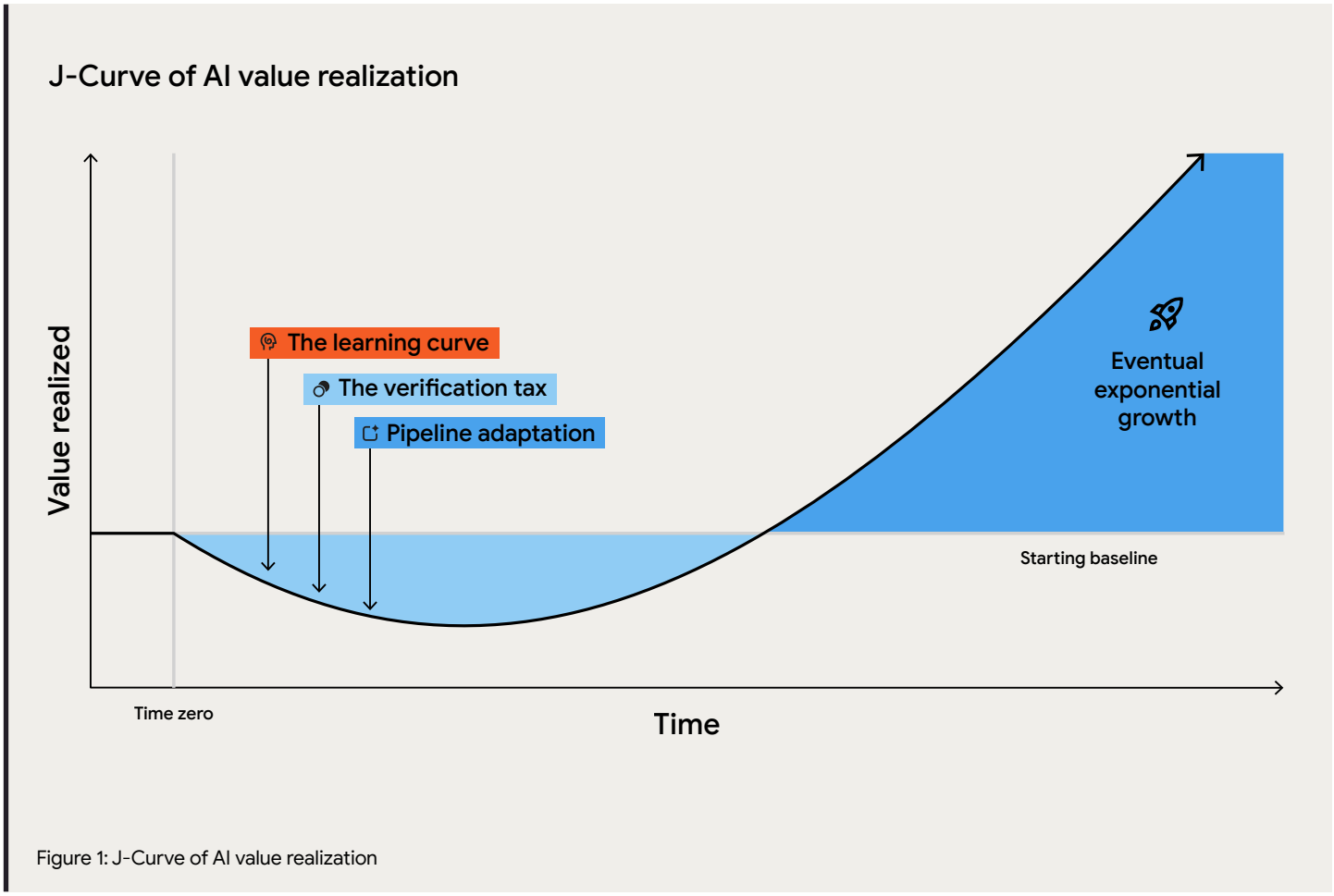


Figure 1: J-Curve of AI value realization

Software delivery is a value engine

Technology is the primary engine for value creation, innovation, and organizational performance. Unlike physical assets, software is not static. Its power lies in its ability to incorporate user feedback and evolve to meet changing needs. Consequently, one of the most critical indicators of a software application's health is how easily, safely, and quickly it can be modified.

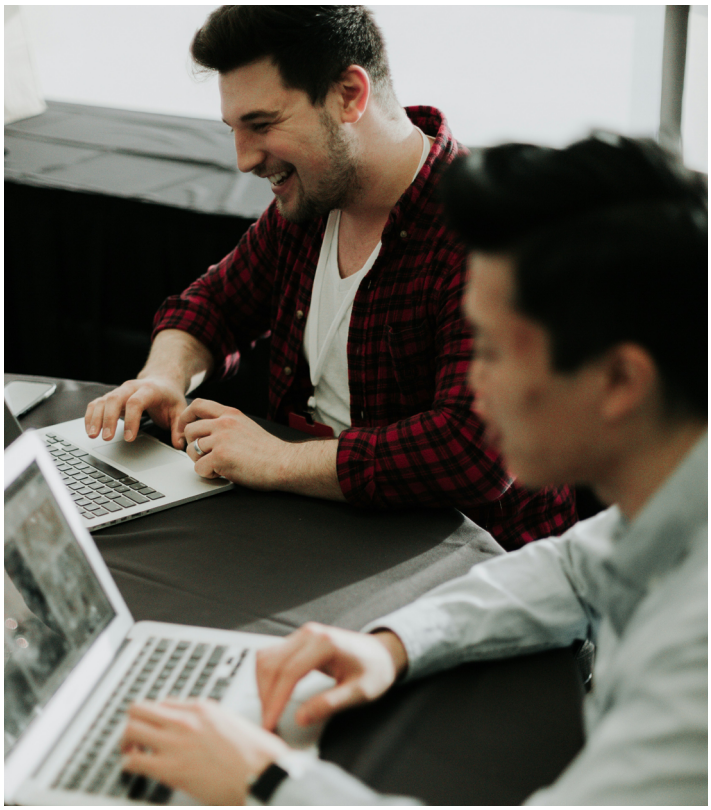
Measuring software delivery performance is essential to forecasting the ROI of AI. It allows us to bridge the gap between engineering metrics and financial impact, ensuring that investments translate into tangible business value.

Track delivery performance to manage risk and velocity

Technology-driven teams need ways to measure performance so that they can assess how they're doing today, prioritize improvements, and validate their progress. DORA's software delivery performance metrics[2] focus on a team's ability to deliver software safely, quickly, and efficiently. They can be divided into metrics that show the throughput of software changes and metrics that show instability of software changes.

- **Throughput:** A measure of the volume and velocity of changes moving through the engineering system. Increasing throughput reduces time-to-market, allowing the business to realize value and recognize returns on new features sooner.
- **Instability:** A measure of the reliability and success rate of software deployments. High levels of instability create operational friction, introduce reputational risk, disrupt revenue streams, and force teams to waste capacity on unnecessary rework.

Low software delivery performance is characterized by low levels of throughput and high levels of instability whereas high software delivery performance is the opposite: high throughput and low instability.



Reduce rework to fund innovation

Delivering software is a value driver, not a cost center. Teams with the best software delivery performance see the highest returns on their investments. These returns are realized not through traditional cost-cutting but by reducing the amount of unnecessary rework. This recovers engineering capacity equivalent to "free headcount," which can be directly reinvested into building new features and driving innovation.

Measuring software delivery performance

Throughput

- Measure of volume and velocity
- Reduced time-to-market
- Realize value sooner
- Recognize returns faster

Instability

- Measure of reliability and change success rate
- Creates operational friction
- Introduces reputational risk
- Disrupts revenue streams
- Waste on unnecessary rework

Understanding performance tiers

High performance:

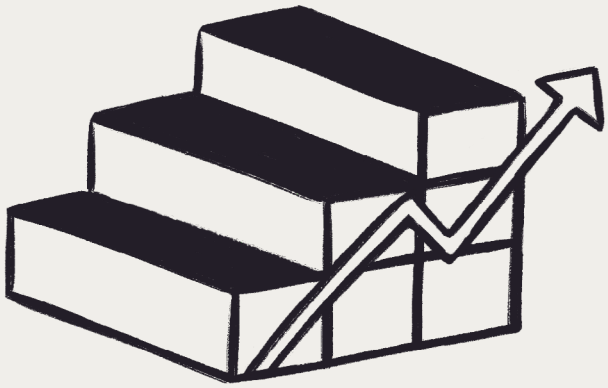
High throughput + low instability

Low performance:

Low throughput + high instability

Start the conversation about ROI

This report is the start of a conversation. It describes a framework that you can use to help drive discussions about both the investments and the returns your team and organization can expect from incorporating AI across the entire SDLC. To assist in modeling this for your organization, we include a calculator with many assumptions that we know need to be fit to your context.



1. T. Winters, T. Manshreck, and H. Wright, *Software Engineering at Google: Lessons Learned from Programming over Time* (Sebastopol, CA: O'Reilly Media, 2020), 167
2. "DORA's software delivery performance metrics." https://dora.dev/guides/dora-metrics

Build the business case for AI investments

DORA's research shows that specific technical and cultural practices predict software delivery performance. When building the business case for AI, this means recognizing that purchasing licenses alone will not guarantee a financial return. Because AI accelerates the speed of development, its impact depends on the health of the environment it operates within.

If your organization possesses mature internal developer platforms and streamlined deployment pipelines, AI will rapidly scale your capacity to deliver user value and drive revenue. However, if your engineering teams are already facing bottlenecks caused by manual testing, heavy bureaucracy, or fragmented data, injecting AI into that system will simply accelerate the accumulation of technical debt and maintenance costs. Therefore, a defensible business case must frame AI not as a standalone solution but as an investment that requires a strong, capable organizational foundation to actually capture the returns.

The software engineering industry has entered the agentic era. This era represents the critical evolution from reactive artificial intelligence tools to autonomous systems capable of independently executing complex, multiple-step workflows. In this paradigm, return on investment is no longer a measure of how many developers an organization can replace. It is a measure of how much latent human creativity can be unlocked by offloading systemic toil to these autonomous agents. We don't measure AI by the code it writes but by the bottlenecks it clears.

Translate technical metrics into financial outcomes

Technology leaders have a unique opportunity to demonstrate the value of their operational expenditures. Board members and financial executives actively seek clear justification for AI initiatives. Measuring the ROI is a mandatory practice to secure ongoing funding and validate engineering strategies. Organizations that possess mature practices for measuring the impact of their AI projects consistently achieve higher business outcomes at an accelerated pace. Leaders must translate technical metrics into financial outcomes. With rigorous measurement, engineering teams build confidence and maintain executive support as their transformation efforts mature.

Budget for the tuition cost of the J-Curve

Measuring these returns accurately requires an understanding of the technology adoption lifecycle. The introduction of AI into software engineering follows a distinct trajectory known as the J-Curve.

The J-Curve is a well-documented phenomenon in major organizational transformations. It illustrates a practical reality: when a team adopts a significant new technology, their performance and productivity temporarily adjust as they learn, before rising to new heights.

The introduction of a new process almost guarantees an initial negative impact on performance, with the depth of the decline directly correlating to the magnitude of the change. Initiatives often fail not because the technology is flawed but because leadership misinterprets this learning phase as a failure and pulls funding during the inevitable dip. We consistently find this trajectory across various technical disciplines, such as continuous delivery and platform engineering. Furthermore, our recent guidance on innovating with generative AI (gen AI) confirms that this identical structural reality applies to the adoption of advanced coding models. The visual representation of this initial investment phase followed by eventual growth looks exactly like the letter J.

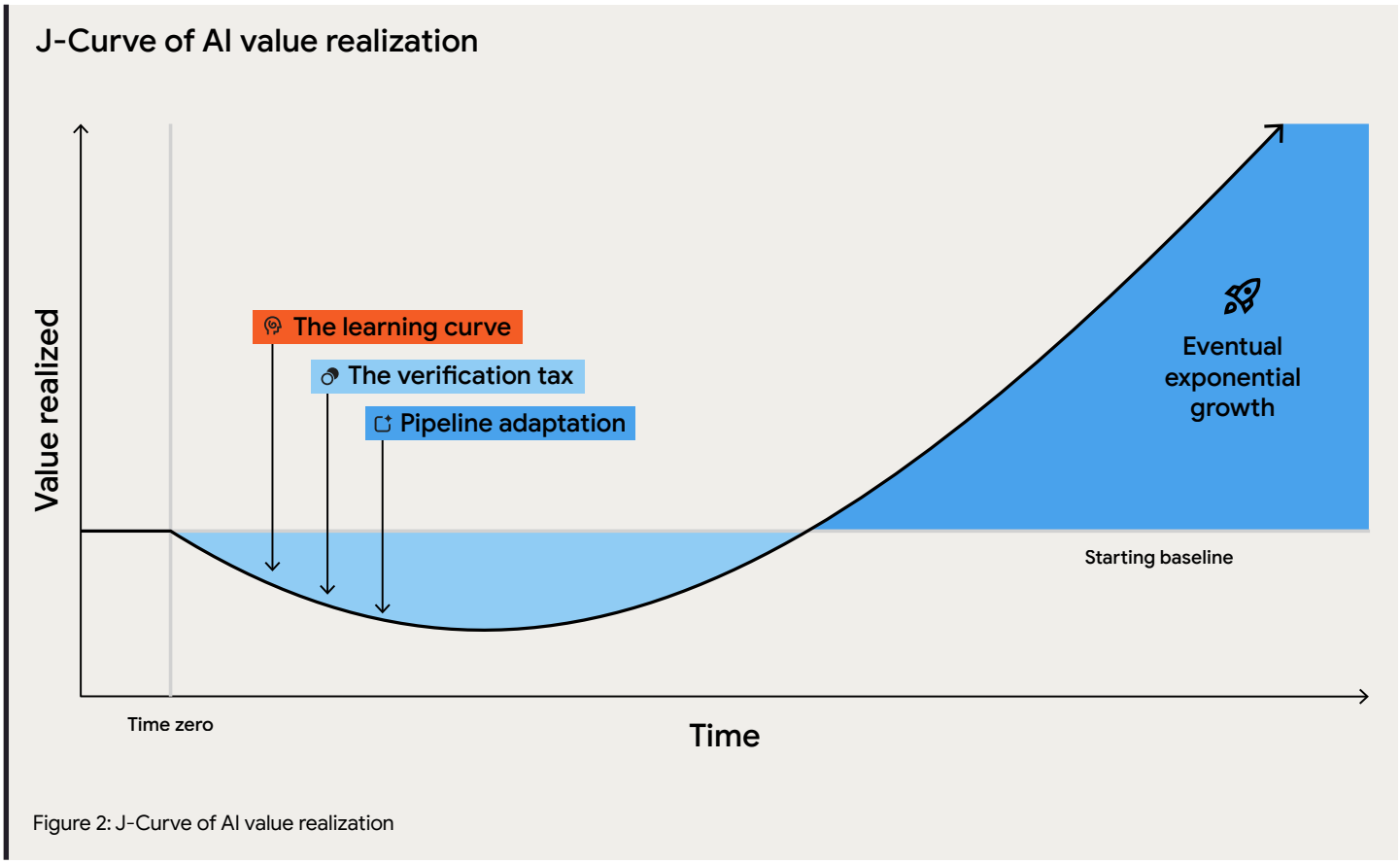


Figure 2: J-Curve of AI value realization

When organizations deploy AI tools, they should expect a productivity dip during the initial phases of adoption. This early adjustment period represents the necessary tuition cost of integration. Several systemic factors contribute to this temporary shift:

The learning curve: Teams dedicate time away from regular feature delivery to learn new interfaces, adapt their daily workflows, and master advanced techniques from users prompting AI to systems built on context, intent, and specification.

The verification tax: Developers invest time reviewing generated code due to concerns about the trustworthiness of output and hallucinations. Furthermore, the tools increase the sheer volume of code produced, which expands the overall review burden required to meet internal security and architectural standards.

Pipeline adaptation: As individual developers generate code faster, downstream processes like testing and change approval must scale to handle the increased volume, revealing opportunities to clear legacy constraints.

While the J-Curve provides a reliable conceptual model for the trajectory of transformation, the specific dimensions of this curve, the depth of the productivity trough, and the duration before growth accelerates remain unpredictable. The AI space is advancing at an unprecedented rate, with generational leaps in capabilities occurring in a matter of months rather than years.

The magnitude and duration of this curve are linked to the unique context of the organization undergoing the transformation. An enterprise's existing technical maturity, its culture of continuous learning, and the baseline health of its internal developer platforms will dictate how quickly it can absorb these rapid technological shifts, mitigate the verification tax, and transition from temporary instability into exponential value realization.

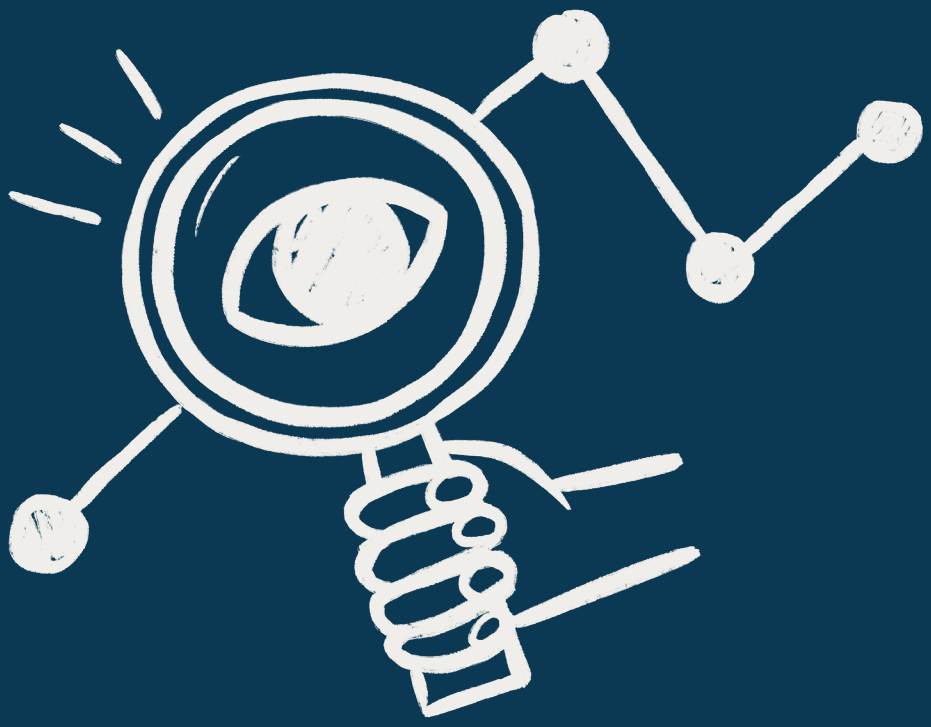
This initial phase is a strategic investment in future velocity. Organizations must explicitly budget for this learning curve to set accurate expectations. DORA capabilities like automated testing[1] and continuous integration[2] are keys to ensuring the dip of the J-Curve rebounds. By understanding the J-Curve, leaders can confidently sustain their initiatives through the adaptation phase, recognizing it as a stepping stone rather than a setback. Strategic patience, backed by accurate measurement and a relentless focus on systemic improvement, ensures that teams successfully navigate the transition and unlock exponential business value.

To smoothly navigate this trajectory, organizations must lean into DORA's software delivery performance metrics framework[3] to better understand the impact of integrating AI into the SDLC. This means looking at lead time for change to assess the impact of agents on velocity, and monitoring change failure rate to understand the efficacy of automated pooling. These metrics provide an operational early warning system that proves the organization is moving through the J-Curve and toward sustainable financial value realization.

It's time to translate these concepts into tangible financial models. As we pivot from theory to practice, we will walk you through the specific mechanics of quantifying value, estimating costs, and determining your final return.

1. "Capabilities: Test automation." https://dora.dev/capabilities/test-automation
2. "Capabilities: Continuous integration." https://dora.dev/capabilities/continuous-integration
3. "DORA's software delivery performance metrics." https://dora.dev/guides/dora-metrics

Understand the market divide on AI returns



There is a common understanding that capturing the financial return of AI is critical. However, the market consensus on AI returns remains divided.

Enterprise adoption is nearly universal, but actual financial impacts vary across organizations. A look at the broader market reveals outcomes spanning from clear value realization to costly operational drag. Understanding these broader trends helps technology leaders explain why similar investments yield different results within software engineering environments.

The positive perspective

Targeted investments backed by strong leadership consistently generate measurable value. According to Google Cloud research,[1] 78% of executives report realizing a return on investment from at least one gen AI use case. Furthermore, 88% of early adopters utilizing agentic AI are already seeing positive returns.[2] When organizations align their technological adoption with clear business objectives and secure executive backing, the financial returns become visible and sustainable. This data indicates that proper governance directly correlates with successful value extraction.

The neutral perspective

The 2025 Stanford Artificial Intelligence Index Report[3] illustrates that expectations for workforce productivity remain highly mixed across different corporate functions. While overall adoption is high, actual structural transformation remains rare across most industries, leading to flat productivity gains for many. Many enterprises observe marginal gains or flat outcomes after initial deployments. This neutral sentiment highlights a critical reality: simply procuring and deploying AI models does not automatically translate into financial benefits.

The pessimistic perspective

Other assessments expose significant friction and risk in enterprise deployments. Research from the MIT NANDA project[4] identifies a divide where internal corporate builds frequently fail, driving workers toward a "shadow AI" economy. Employees often leverage unauthorized consumer applications to achieve the daily productivity gains that official enterprise platforms fail to deliver. While these unauthorized tools provide immediate localized returns, they introduce security risks and fail to scale systemic value across the business. The primary barrier to successful adoption is organizational design rather than budget constraints or technology integration challenges.

Treat AI as an amplifier of your existing engineering systems

These market observations align with the core thesis of our 2025 DORA research: AI functions primarily as an amplifier within the software development lifecycle. It magnifies the existing strengths of highly effective teams while concurrently exacerbating the dysfunctions of struggling organizations.

When companies see mitigated or negative returns from their engineering investments, the failure rarely stems from the technology itself. The structural flaw is often the absence of a robust underlying organizational system. Without a solid foundation built on quality internal platforms and clear workflows, AI merely generates isolated pockets of productivity.

For example, a developer might write code significantly faster using an automated assistant, but that code simply piles up in front of manual security reviews or brittle deployment pipelines. Those localized gains are lost to downstream chaos, erasing any potential return on investment.

To bridge the gap between market promises and actual engineering returns, organizations must treat adoption as a systemic organizational transformation rather than a simple tool procurement exercise. An internal research study conducted by Google in 2025 drew on data from global enterprises that have successfully accelerated their returns.

This study revealed a clear blueprint for the transformation, identifying five strategic priorities successful leaders embrace to maximize value:

Align strategy from the top: Unify business and AI strategy, ensuring adoption is championed directly by the executive team.

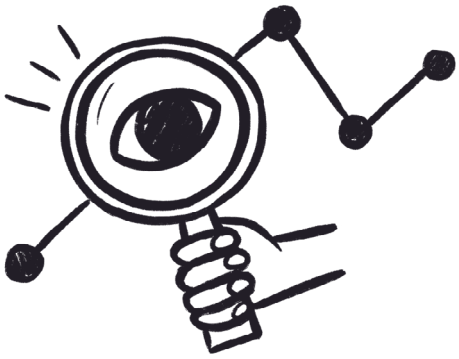
Elevate data quality: Enhance the quality of internal data and knowledge management systems.

Prioritize by value: Estimate the business value for individual use cases, prioritizing them according to their potential financial impact and ease of implementation.

Invest in human capital: Upskill existing staff, hire specialized talent, and cultivate the right external partnerships.

Optimize infrastructure: Provide the tooling and compute resources required to efficiently sustain and scale these advanced models.

Organizations that commit to these five structural requirements move beyond the stagnation seen in the broader market, turning localized engineering speed into sustainable financial growth.



1. Google Cloud, *The ROI of AI 2025: How Agents Are Unlocking the Next Wave of AI-Driven Business Value* (2025), https://cloud.google.com/resources/content/roi-of-ai-2025
2. Google Cloud, *The ROI of AI 2025*
3. "The Stanford University Institute for Human-Centered Artificial Intelligence, *The 2025 AI Index Report,* (2025), https://hai.stanford.edu/ai-index/2025-ai-index-report
4. MLQ.ai, *The GenAI Divide: State of AI in Business 2025* (2025), https://mlq.ai/media/quarterly_decks/v0.1_State_of_AI_in_Business_2025_Report.pdf

DORA COMMUNITY

The DORA Community[1] provides a platform for professionals to engage with this research and apply it to improve their own organizational performance.

Why join the DORA Community?

There are several reasons why you should be a part of the DORA Community:

Learn from experts and peers: The Community offers opportunities to learn from guest speakers and other members through presentations and discussions.

Stay up to date with research: Be the first to know about new information and publications from DORA.

Collaborate and discuss: The DORA Community Google Group[2] provides a forum for asynchronous conversations, announcements, and event invitations. This allows members to discuss topics and share their experiences with others in the field.

Engage in Community events: A calendar of events, both virtual and in-person, is available on DORA.community.

Contribute to the conversation: Contribute to the conversation by listening, talking, and participating in chats. The Community values the input of its members and provides a space for ongoing discussions on topics like leadership, team empowerment, and the evolution of technology practices.

1. The DORA Community, https://dora.community
2. The DORA Community Google Group, https://groups.google.com/g/dora-community/about

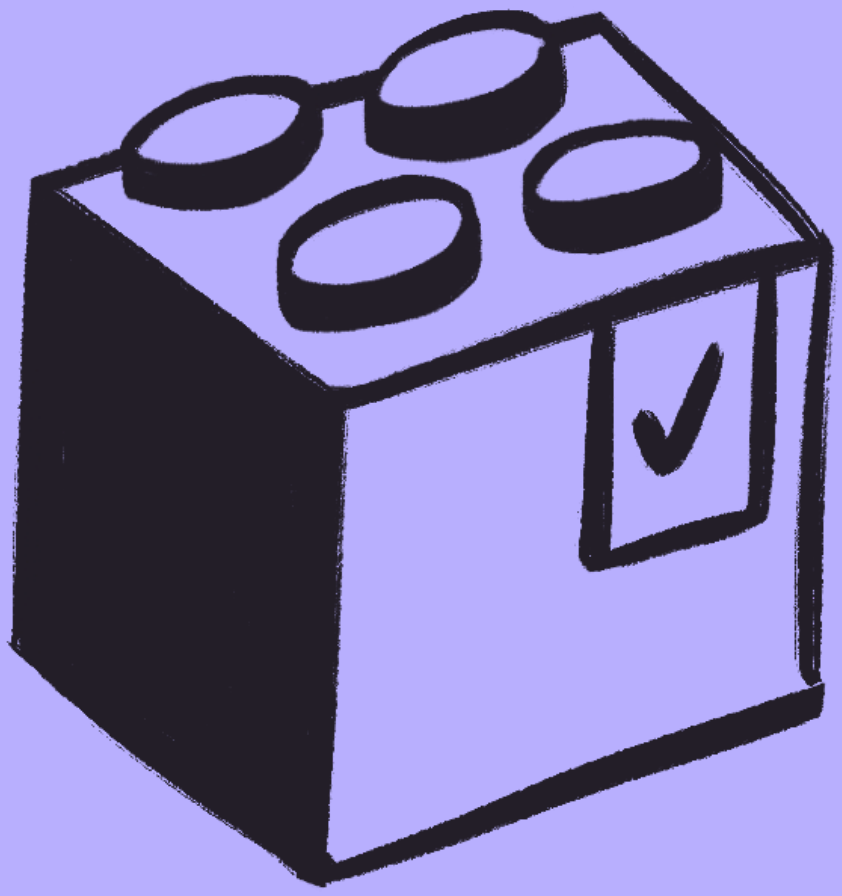
Calculate the ROI of AI-assisted software development

Turning engineering speed into sustainable financial growth requires a rigorous approach to measurement. As outlined in "Understand the market divide on AI returns," organizations must prioritize their adoption strategy based on actual business value. To execute this successfully, leaders must define and calculate their return on investment.

In this chapter, we will break down the specific value drivers that fuel your return on investment.

Return on investment is the ratio between the benefits from an investment and its cost.

$$\text{ROI (\%)} = \frac{\textit{Value} - \textit{Investment}}{\textit{Investment}}$$



The value of AI-assisted software development

In this section, we will look at the value from the AI investment before we come to cost and bring it all together in ROI.

Value framework

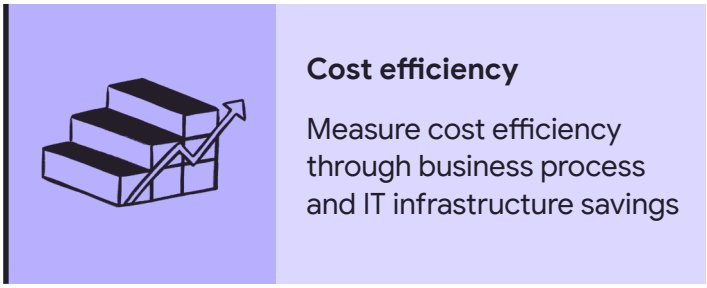


Figure 3: Google Cloud's value framework for AI-assisted software development[1]

When organizations evaluate the return on AI investment in software development, they typically begin with the most quantifiable benefit: cost efficiency. This encompasses direct reductions in software development costs, IT infrastructure spend, and the operational overhead of broader IT processes, but it also captures efficiency gains that ripple downstream into business functions. Cost efficiency is a strong early signal of AI's value, but only when the surrounding conditions including tooling, workflows, and organizational readiness are aligned.

Improved productivity is what most AI solutions target as a benefit today. These improvements apply to both IT departments and broader business operations. Sometimes, productivity improvements translate into cost savings when process cost is cut (for example by reducing the need for external detection services or other outsourced work). Most often, productivity improvements free up resources for higher-value work, such as relieving developers from routine tasks, giving them time they can invest in new developments.

AI solutions are also aiming to improve developer experience. By reducing repetitive work, AI tooling can increase developer engagement, lower cognitive load, and improve retention, driving outcomes that carry real financial consequences even when they resist direct measurement.

Developer experience should be assessed through survey data and attrition metrics rather than productivity proxies alone.

Faster development cycles, higher product quality, and more responsive iteration translate into better software for end users, which in turn improves user satisfaction and strengthens brand loyalty. Attribution remains a challenge here, as user satisfaction responds to many variables simultaneously, but net promoter score (NPS) tracking and cohort analysis can establish credible linkage to engineering practice.

Finally, AI is expected to impact business growth—growing leads, conversion, and ultimately revenue for the organization. AI investment feeds developer output, which shapes product quality, which influences user retention, which eventually drives business growth. Every step adds measurement complexity but it's also the reason executives sign off on AI investments in the first place.

Together, these five pillars make up what we call cumulated business value. The further right you move in the framework, the less direct the connection to AI-augmented engineering, and the harder it becomes to measure.

In the subsequent sections, we want to show how engineering teams work to contribute to each of these pillars of value and how AI can amplify this for better and for worse.

Drive business growth through accelerated innovation

Let's start with the end in mind. Engineering teams grow the business with the help of AI when they deploy new features, products, and services that users pay for, so they generate revenue for the business.

AI enables engineers to accelerate development and brings previously impossible products within reach. By leveraging AI to refine existing systems and build novel features, teams directly drive organizational growth.

Improve user experience to lead financial growth

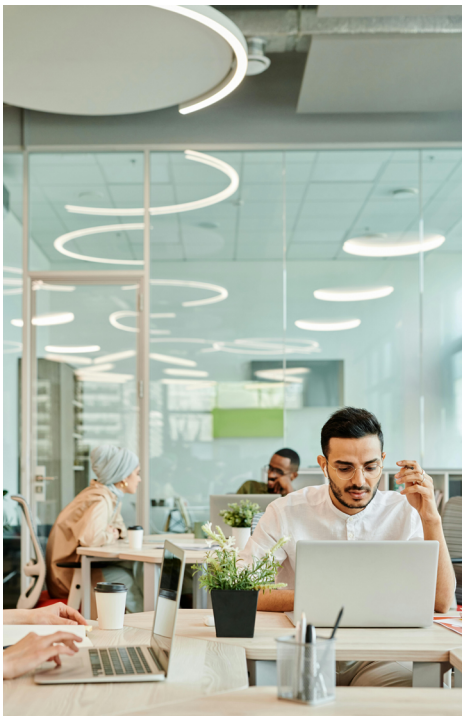
Software that effectively solves real user problems and removes friction from their workflows naturally earns ongoing usage and loyalty. User experience is a leading indicator for business growth, and much easier for engineering teams to impact.

User-centric focus[2] is one of the capabilities identified in the DORA AI Capabilities Model.[3] A relentless focus on the user—and regular interaction with users—ensures engineers are creating value for users and hence for the business.

Boost developer experience to reduce turnover costs

One of the aims of using AI in software development is to make developers happier. AI can take care of repetitive tasks, documentation, and other work that developers often don't enjoy. When engineers appreciate the help of these tools and have a more fulfilling working experience, they will be more energized, motivated, creative, and productive to create products that delight users and grow the business.

This effect was confirmed in the 2025 DORA report, which found that more than 80% of survey respondents reported a perception that AI had increased their productivity.[4] AI adoption was also shown to increase individual effectiveness, as well as the amount of time an individual spent doing work they felt was valuable and worthwhile.[5]



Increase developer productivity

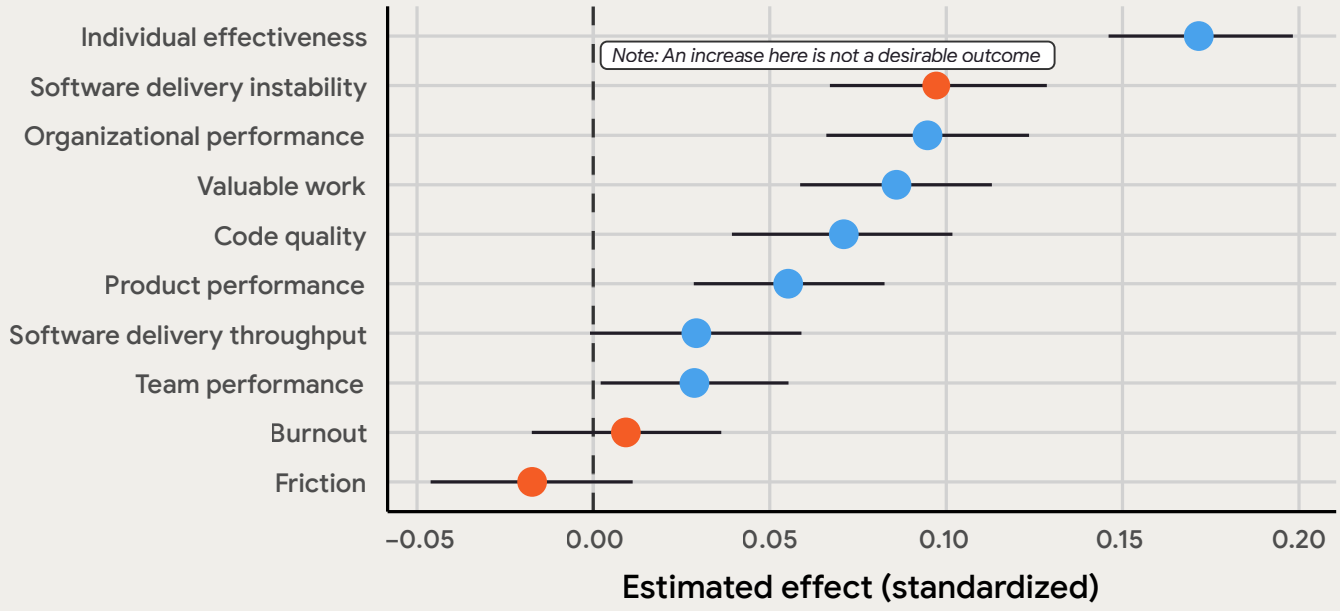
Improving developer productivity is one of the top ambitions of AI in software engineering. The 2025 DORA report found a positive impact of AI on productivity in several ways.

- Increased AI adoption was associated with an increase in individual effectiveness—this was the strongest effect, compared with all other outcomes of AI adoption that were studied.
- Increased AI adoption was associated with an increase in code quality. Better quality code leads to more productive processes because less rework is required.[6]
- Increased AI adoption was associated with an increase in higher software delivery throughput. Teams are able to deliver more software changes in less time.[7]
- Finally, increased AI adoption was associated with an increase in positive impact on team performance. Under team performance, different aspects were evaluated—delivering innovative solutions, adapting to change, effectively collaborating with each other, being able to rely on each other, efficiently working together—which are all components of team productivity.[8]

It's important to note that in the 2025 DORA report, increased AI adoption was associated with an increase in software delivery instability.[9] AI-assisted coding can increase the volume and velocity of code generation, overwhelming existing deployment pipelines and manual review gates.
We view this temporary rise in instability as part of the expected J-Curve (or tuition cost) of early AI adoption. To flatten this curve and safely manage AI-accelerated development, organizations must invest in their underlying engineering platforms,[10] emphasizing strong version control practices[11] and small batch sizes[12] to catch errors before they reach the user.

The landscape of AI's impact

Estimated effect of AI adoption on key outcomes, with 89% credible intervals



For outcomes in orange (such as burnout), a negative effect is desirable.
Figure 4: Standardized estimated effects of AI adoption

While there is an expectation that AI adoption will reduce burnout and friction—thereby boosting productivity—the 2025 DORA research suggests otherwise. The data indicates that workplace friction is a complex issue extending far beyond the automation of rote tasks. For an individual developer, AI does not necessarily make friction vanish; rather, the friction moves. The effort saved from manually writing boilerplate code is often replaced by a verification tax—the cognitive load required to iterate on prompts and rigorously audit AI-generated code that looks remarkably similar to correct code.

Similarly, burnout remains stubbornly resistant to purely technological solutions.[13] If developers use AI to move faster but are still nested within a chaotic system featuring brittle tests and heavy bureaucracy, the technology simply amplifies the surrounding organizational friction.

Ultimately, these effects appear to be driven by the systems, processes, and culture in which an individual is nested rather than by the technology itself.[14]

Despite these caveats, the positive impact of AI on software development productivity is probably the best confirmed effect so far.[15, 16, 17] We just do not want to stop there. Improved productivity should translate into better developer experience, better user experience, and ultimately quantitative business growth.

Drive cost efficiency and reinvest in more strategic initiatives

When we think about AI improving productivity, we also expect AI to help reduce cost—both in IT and in business processes. However, this happens only when AI is replacing work that is not needed anymore.

In software development, the effort that is saved with AI is reinvested into more productive or innovative development work, so we will see no direct cost savings at first sight. A bit further downstream, however, better developer productivity can lead to cost savings. When developer resources are reinvested into more innovative work, organizations do not have to hire additional resources to do that work. This is cost avoided.

Also, when developers are happier at work, they are less likely to leave. This reduces the cost to rehire, which is significant because developers are a scarce resource.

A direct source of cost savings may be the reduction of IT infrastructure cost. AI-assisted software development can help to use infrastructure resources more efficiently and hence reduce infrastructure cost (for example, cloud consumption).

Significant cost savings in the downstream business processes (for example in HR or production processes) are unlocked with the AI products that developers create. It is therefore a relevant effect, but as the focus of this report is software development, we do not pursue it further.

Improved cost efficiency is a benefit in itself (and often the priority for AI implementations), but it ultimately also helps to grow the business—by freeing up a budget to invest in growth.[18]

In "Identify the value drivers," we want to tie up these effects into a model that allows us to map and calculate the ROI from AI-assisted software development.

1. This is an updated version for AI, based on the value framework for cloud: U. Löbbert-Passing, D. Murphy, and S. Sarma, *Leveraging Cloud FinOps to Measure the Business Value Realized by Your Cloud Transformation* (white paper, 2022), https://services.google.com/fh/files/misc/leveraging_finops_to_measure_the_business_value_realised_by_the_cloud_transformation.pdf
2. "Capabilities: User-centric focus." https://dora.dev/capabilities/user-centric-focus
3. DORA, *DORA AI Capabilities Model* (2025), https://dora.dev/dora-aicmr
4. DORA, *2025 State of AI-assisted Software Development* (2025), 30, https://dora.dev/dora-report-2025
5. DORA, *2025 State of AI-assisted Software Development*, 37–38
6. DORA, *2025 State of AI-assisted Software Development*, 30
7. *DORA, 2025 State of AI-assisted Software Development*, 37–38
8. *DORA, 2025 State of AI-assisted Software Development*, 37–38
9. *DORA, 2025 State of AI-assisted Software Development*, 41
10. "Capabilities: Platform engineering." https://dora.dev/capabilities/platform-engineering
11. "Capabilities: Version control." https://dora.dev/capabilities/version-control
12. "Capabilities: Working in small batches." https://dora.dev/capabilities/working-in-small-batches
13. DORA, *2025 State of AI-assisted Software Development*, 38–39
14. DORA, *2025 State of AI-assisted Software Development*, 39–40
15. K. White, "Key Insights from Our Inaugural Survey on the ROI of AI in the Public Sector," 2026, https://cloud.google.com/blog/topics/public-sector/key-insights-from-our-inaugural-survey-on-the-roi-of-ai-in-the-public-sector
16. Google Cloud, The ROI of Gen AI (2025), https://services.google.com/fh/files/misc/the_roi_of_generative_ai.pdf
17. Stanford University Institute for Human-Centered Artificial Intelligence, *The 2025 AI Index Report* (2025), https://hai.stanford.edu/ai-index/2025-ai-index-report
18. J. Haughwout, "Quantifying the Impact of Developer Experience: Amazon's 15.9% Breakthrough," 2025, https://aws.amazon.com/de/blogs/enterprise-strategy/business-value-of-developer-experience-improvements-amazons-15-9-breakthrough

Identify the value drivers

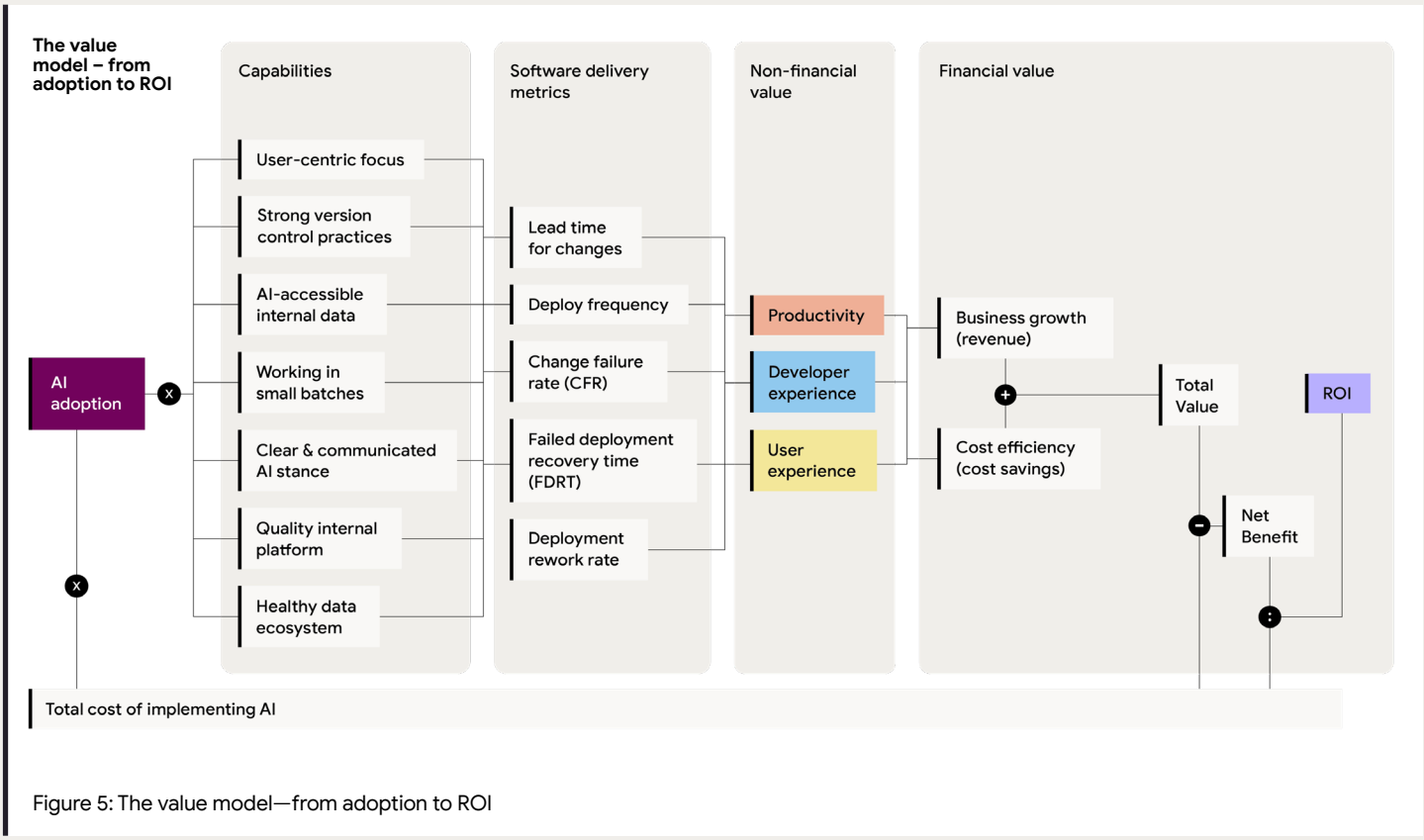


Figure 5: The value model—from adoption to ROI

In the "Calculate the ROI of AI-assisted software development" chapter, we outlined how developers, assisted by AI, impact business value. We want to map these chains of effects into a model that ultimately allows us to calculate the ROI from AI-assisted software development.

There are lots of other ways that an organization can leverage AI. This model and report are primarily focused on the effect that AI has in the software development lifecycle.

The input to the model is AI adoption in the SDLC. AI adoption also determines the cost of the AI tools that we need to calculate ROI (see the "Quantify your AI investment" chapter).

Our approach builds directly upon the DORA AI Capabilities Model report.[1] The model highlights seven crucial capabilities that help transform basic AI adoption into actual business value. These capabilities provide the necessary guardrails and workflows to ensure that the localized speed and productivity gains from AI are not lost to downstream bottlenecks or instability.

The model works like a flow from left to right. Starting from the left-hand side of the model, AI adoption needs to be underpinned by capabilities in order to improve DORA key metrics. Improvements in the key metrics manifest in non-financial value such as higher productivity, improved developer experience, or better user experience before they hit revenue or cost. That's why we speak

about the first three as leading indicators, and cost savings and revenue as lagging indicators.

Given our initial ROI formula,

$$\text{ROI (\%)} = \frac{\textit{Value} - \textit{Investment}}{\textit{Investment}}$$

we add new revenues and cost savings to obtain the total value of AI-assisted software development. From the total value, we deduct the cost of implementing AI—which is the investment—to give us the net benefits. We then divide the net benefits by the cost of implementing AI to give the ROI percentage (for example, 39%).

Quantify value

To calculate value, we are using a simplification of the business case approach that Google Cloud's Value Realization practice has been using with customers over many years to calculate the business value from IT investments.

In this section, we will explain how we calculate value.

When we are doing this, it is important not to double and triple count values. For example, when your developers save an hour a day, you can use that hour to avoid hiring more developers. You can also use it to ship more products and generate revenue. In our model, we assume that this is not an "either/or" because reality is not as clear cut and we want to give room for different effects. When quantifying the effects and making assumptions, we need to keep this in mind.

Note on methodology

Treat these calculations as a high-uncertainty estimate meant to spark a conversation, rather than a rigid mathematical formula. As the saying goes, all models are wrong,[2] but we hope this one proves useful for your team as you assess the impact of AI in the software development lifecycle.

As with any DORA insights, the real value comes from contextualizing the findings and applying them within your own organization. We've provided the ROI of AI-assisted software development calculator[3] so you can explore the mechanics, adjust the assumptions to match your reality, and build your own estimate. Once you've had a chance to experiment, we invite you to share how it's working out for your team in the DORA Community.[4]

Measure direct productivity gains

We start with calculating the effect of AI in software development on productivity because it's the most direct.

Quantify developer time saved

When we think about how AI adoption, boosted by the capabilities, influences the DORA key metrics, we have to calculate two types of effects:

- **The positive effects:** Improvements in lead time for changes, deployment frequency, and failed deployment recovery time, which reflect higher software delivery throughput. AI outcomes that may contribute to the change are better individual effectiveness, better team performance, better code quality, and reduced friction.[5]
- **The negative effect**: A deterioration in the key metrics, so lower productivity, from higher software delivery instability.

The change in productivity will then be mapped to an impact on revenue and cost.

To quantify the effects of AI in software development on productivity, we are estimating:

1. The time saved by individuals by using AI—which we are translating into avoided cost for additional headcount. When our developers are more productive, we can reinvest the freed-up capacity into more valuable work and we don't need to hire additional developers.
2. The revenue from extra feature deployments. When our developers are more productive, we can ship more features to market faster and generate additional revenue for the company.

Factor in the "instability tax"

When estimating time saved, we need to take into account what the DORA research found to be a negative effect of adopting AI: an increase in software instability.[6] More unstable software is likely to detrimentally impact developer productivity because developers have to spend additional time on dealing with this increased instability.

To give you an impression of the size of this effect: the DORA research found that AI adoption has different effects on the various outcomes, such as individual effectiveness and code quality—some are higher, some lower (see Figure 4 in the "Calculate the ROI of AI-assisted software development" chapter). The largest effect was measured on individual effectiveness. The effect on instability was second largest, larger than the impacts on organizational performance, code quality, and so forth.[7]

This also indicates that developers spend less time on greater software instability than they gained on individual effectiveness in the first place.

Convert reclaimed hours into cost savings

Crucially, we are not assuming that when we save time in software development we are saving the cost for this development effort. We strongly recommend organizations do not adopt a headcount-reduction strategy, which has a negative impact on morale and organizational culture, can reduce efficiencies, and can even incentivize workers to not improve their work processes. Instead, this effort should be reinvested into new, innovative, or more productive work.

Retaining and training existing talent is more cost-effective, preserves institutional knowledge, and gives organizations an advantage by having a strong technical workforce that is engaged and continuing to learn. This is a huge win as the costs of turnover far outstrip costs of retaining talent.

What we think we do save is the cost of hiring additional developers: That is:

> **Headcount reinvestment capacity = Staff size * Fully loaded salary * Net time saved per developer**

In our example, if we assume a fully loaded salary of $176k[8] (please adapt according to your country, on-/offshoring ratio, and other factors), that gives us $11M of savings.

Project revenue from accelerated feature delivery

To estimate revenue from extra feature deployments, we make assumptions on:

- **How many more features we can deploy** (the calculator asks you for target versus current number of features deployed). Features are user-facing enhancements designed to improve the product experience.
- **The idea success rate.** This is the percentage of shipped features that increase revenue for the application. Research suggests it's about one third.[9]
- **The average revenue impact per successful feature.** We recognize this is likely very difficult to project. In the calculator, we recommend a conservative value between 0.01% and 1%.
- **The product portfolio revenue.** This is the annual revenue driven by the software in scope.

We then calculate the revenue from extra feature deployments as:

> **(Target features deployed – Current features deployed) * Idea success rate * Revenue impact * Portfolio revenue**

For example, for a company with a portfolio revenue of $100M, six additional features per year, each with a 33% success rate and 0.5% revenue impact, that would give almost $1M of additional revenue.

User experience

The next value impact we are calculating is related to user experience.

The DORA research found that AI adoption, coupled with the capabilities, produces better product performance. A better performing product will make users happier.

This is likely to have a revenue impact because happy users buy more. It is a relatively loose link though, so we are not pursuing it in our calculator.

Instead we focus on additional revenue or costs avoided, such as reputational damage, due to reduced downtime of the applications.

Reach out to us to go beyond the calculator

We wanted to keep things simple and get an initial ROI estimate into your hands quickly so that you can start a discussion. That's why our calculator does not include all the positive and negative impacts of using AI in the SDLC, only the major ones.

If you want to move beyond these and take a deeper look into your specific situation, Google Cloud Consulting's delta team[10] is here to help you.

Please note that the outcome of this calculation may also be negative due to the "instability tax." In that case, revenue is lost and / or additional cost incurred due to additional downtime.

To calculate this effect, we need to make additional assumptions on:

- **The current and target change failure rates (CFR):** The current rate is the percentage of changes to production or released to users that result in degraded service today. The target rate is what you expect it to be when you run on AI. Due to the "instability tax," the target can also be higher than the current.
- **The failed deployment recovery time (FDRT):** The average number of hours it generally takes to restore service after a change to production or release to users results in degraded service.
- **The revenue lost / cost of downtime per hour:** The cost of one hour of system outage. Estimate revenue lost and any other costs, such as reputational damage.

The additional revenue or costs avoided by reducing downtime is then:

> **(Current deploys * Current CFR * FDRT * Cost of downtime per hour) – (Target deploys * Target CFR * FDRT * Cost of downtime per hour)**

For example, if we assume:

- Current number of deployments per year = 50
- Current CFR = 5%
- FDRT = 4 hours
- Cost of downtime per hour = $100k
- Target number of deployments per year = 56
- Target CFR = 6%

this gives us an additional cost (negative value) of $344k—because we assumed the CFR would go up.

To keep it simple, we are not including the effect of improved deployment rework rate in our calculator. It's an alternative route to calculate the additional revenue or costs avoided by reducing downtime.

Boost developer experience to retain talent

AI adoption that improves team performance, reduces friction, and boosts individual effectiveness naturally leads to happier developers. Conversely, unmanaged software instability can cause frustration.

When developer experience improves, turnover drops. Given that replacing a developer can cost 1.5 to 2 times their annual salary (including recruiting, onboarding, and lost productivity), retention is a massive financial lever.

Note: Because this link is highly variable, we have excluded it from the baseline calculator to keep your estimates conservative. However, it serves as a powerful qualitative argument when presenting your business case.

Consolidate your total projected value

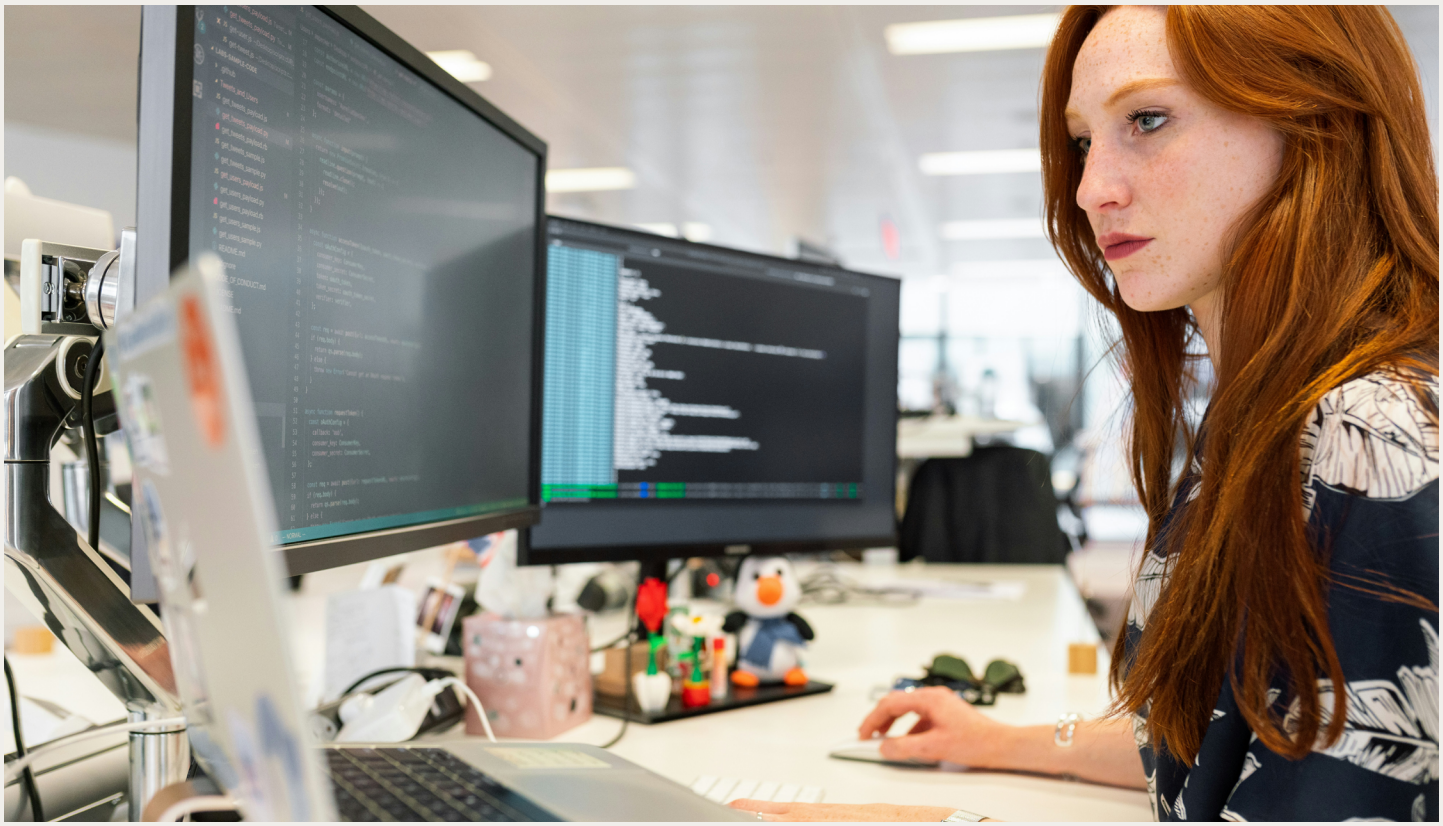
Our last step consists of adding up all the impacts on cost and revenue (positive and negative) to yield the total value we are expecting from AI-assisted software development.

The calculator does this automatically for you:

> **Total value = Headcount reinvestment capacity + Revenue from extra features + Revenue or costs avoided by reducing downtime**

In our example, we receive approximately $11.6M of value per year.

Please note that this number is just an example. From all the above, you will have seen how many



assumptions we made to come up with this number. If you change one of them, the total value estimate will change. If you change all of them, the total value estimate will change dramatically.

The point here is to provide a schema with which you can calculate value yourself.

We also want to highlight the levers you have in software development to impact financial value. Don't stop at thinking of the impact of AI in your software development teams as improvements in individual effectiveness or code quality only, for example. You can start looking at early indicators today, as described in the "Map your AI investment roadmap" chapter. This report and the calculator[11] help you to translate these improvements into higher productivity, better developer experience, augmented user experience, cost savings, and ultimately quantitative business growth.

We also acknowledge that the different levers of value we calculated are of differing "solidity." Tying an increase in deployments to a revenue

lift is speculative and may be challenged by a CFO. Of course, it is safer to focus the value strictly on engineering efficiency and infrastructure savings. If you want to exclude value levers you can always do so in the calculator by "zeroing out" the respective assumptions.

We think a value estimate is a door opener for a discussion. We explicitly do not want to restrict value to the very hard cost savings. It is important for developers to know that they are also impacting user satisfaction and revenue, even if the links are weaker.

We will now need to compare the total value we calculated against the cost of implementing AI-assisted software development, to get to ROI.

1. DORA, *DORA AI Capabilities Model* (2025), https://dora.dev/dora-aicmr
2. "All models are wrong." https://en.wikipedia.org/wiki/All_models_are_wrong
3. "DORA ROI of AI-assisted software development calculator." https://dora.dev/ai/roi/calculator
4. "DORA Community." https://dora.community
5. See the DORA AI Capabilities Model report for a detailed model of how the capabilities impact outcomes
6. DORA, *2025 State of AI-assisted Software Development* (2025), 37–42, https://dora.dev/dora-report-2025
7. DORA, *2025 State of AI-assisted Software Development*, 38
8. Howdy.com, "A CTO's Guide to the True Cost of a US Developer," 2025, https://www.howdy.com/for-ai/true-cost-us-developer
9. N. Larsen et al., "Statistical Challenges in Online Controlled Experiments: A Review of A/B Testing Methodology," 2023, https://arxiv.org/abs/2212.11366
10. "Contact us." https://dora.dev/ai/roi/contact
11. "DORA ROI of AI-assisted software development calculator." https://dora.dev/ai/roi/calculator

Quantify your AI investment

To realize ROI, organizations must invest in the underlying engineering systems and plan for the J-Curve of value realization.

A realistic cost model must look beyond the price of software licenses to include infrastructure, enablement, and the temporary disruption to existing workflows. Most organizations adopting AI will encounter the J-Curve. In financial terms, this phase represents the "tuition cost" of transformation. Factoring this operational expense into your budget upfront creates a realistic baseline for your financial model and protects the initiative from premature scrutiny.

To assist in modeling this for your organization, we include a calculator with many assumptions that we know need to be fit to your context. Our framework divides the financial impact into two main categories: total hard costs and the J-Curve cost. Total hard costs encompass the direct capital expenditure (CapEx) and operating expenses (OpEx) necessary to secure tools and scale underlying platforms. The J-Curve cost realistically models the productivity dip, accounting for the initial learning curve, workflow friction, and the verification tax required to review generated code. By explicitly forecasting this cost and investing in the underlying engineering systems, leaders can build an accurate, comprehensive budget that accounts for both the visible expenses and the hidden realities of AI adoption.

Calculate direct hard costs for tooling and infrastructure

When forecasting the ROI of AI-assisted software development, you must account for the direct financial outlays required to support your new capabilities. These investments span both CapEx and OpEx.

- **Licenses and usage fees:** The most visible expense is often the price of the tools themselves. To build an accurate model, define the inputs for the annual AI license cost per user, which represents your base subscription. Also account for the additional annual AI costs per user, which cover variable expenses such as API usage or token costs.
- **Infrastructure investments:** Annual AI infrastructure costs include the compute, networking, storage, and monitoring capabilities required to support these tools safely and effectively.
- **Enablement:** Factor in the training costs. Investing in training and change management is critical.

Please see the calculator for additional instructions on how to estimate these investments with a few assumptions.

By summing these individual elements—licenses, infrastructure, and enablement—you can accurately calculate the total hard costs (tooling and training) for your organization. This figure provides the baseline investment required before factoring in the productivity impacts of the J-Curve.

> **Direct hard costs = ((License cost + Additional AI costs + Training cost) * Staff size) + Additional infrastructure costs**

The sample ROI calculator uses the following estimates:

- **Annual AI license cost per user:** $250
- **Additional annual AI costs per user:** $80
- **Annual training costs per user:** $9,600
- **Technical staff size in number of full-time employees (FTE):** 500
- **Additional annual AI infrastructure costs:** $100,000

These estimates yield $5,065,000 as the calculated value for the total hard costs (tooling and training).

Forecast the J-Curve and the tuition cost

To accurately model this tuition cost in your financial framework, you must estimate two key variables. First, determine the J-Curve productivity drop, which is the percentage of engineering capacity temporarily lost to new workflows and friction. Next, define the timeline, forecasting how many months this period of disruption is expected to last.

The calculator uses these inputs to generate the J-Curve cost. This output calculates the direct financial equivalent of your lost productivity by multiplying the anticipated percentage drop by the timeline and applying that reduction to your technical staff size and their fully loaded salaries. Quantifying this tuition cost prepares you for transparent conversations with finance leaders, helping them understand that the initial dip is a planned phase of systemic investment, not a permanent loss of velocity.

$$\text{J-Curve cost} = \text{Staff size} * \text{Salary} * \text{J-Curve drop} * \frac{\text{J-Curve duration}}{12}$$

The sample ROI calculator uses the following estimates:

- **Technical staff size in number of full-time employees (FTE):** 500
- **Average fully loaded technical staff salary:** $176,000
- **J-Curve productivity drop:** 15%
- **J-Curve productivity drop timeline:** Three months

These estimates yield $3,300,000 as the calculated value for the J-Curve cost.

Identify additional and indirect costs

The ROI model captures direct financial outlays and the immediate productivity dip of the J-Curve. A more comprehensive cost model might also account for the indirect and ongoing operational costs associated with AI adoption. These expenses may not appear as direct line items in a one-year snapshot but will manifest elsewhere in your system's performance metrics.

Verification tax

AI-assisted software development naturally increases team velocity and the volume of generated code, which brings more pressure on the code review process.[1] This "verification tax" represents the extra time invested in automated testing or synchronous code reviews required to handle the higher volume of outputs.

This is not modeled as a direct cost but system metrics like deployment frequency serve as a proxy. The higher the verification tax, the fewer annual software deployments will be possible. Similarly, the lead time for changes will increase because of this heightened verification burden.

Teams can compensate for this risk[2] by implementing technical guardrails, such as nonoptional checkpoints and pre-commit hooks paired with static analysis to catch known vulnerabilities. Additional opportunities to counter these costs include investing heavily in automated testing,[3] leveraging AI to assist with code reviews, and providing better context to the AI[4] to improve initial code quality.

Ongoing maintenance and monitoring

Keeping an AI solution up to date, secure, and performant requires continuous human intervention. This includes monitoring the models for drift, managing API updates, and continuous data retraining.

Neglecting these ongoing maintenance routines will likely surface as a negative impact on software delivery metrics[5] such as change failure rate and failed deployment recovery times.

Normalizing process guardrails—such as working in small batches[6] and maintaining strong version control practices[7]—ensures the velocity offered by AI does not push teams off target. Upholding cultural routines like mandatory synchronous security reviews and architectural decision records also helps maintain the team's familiarity with the codebase, preventing legacy code rot.

Determine your total first-year investment

To fully understand the financial commitment required for AI-assisted software development, you must combine the visible financial outlays with the hidden costs of temporary disruption. By adding your total hard costs—which cover your tooling, infrastructure, and enablement—to the anticipated J-Curve cost, you arrive at your total one-year investment.

This comprehensive investment figure is what you'll weigh against the recovered capacity and business value generated by the AI multiplier. This initial investment sets the foundation for accelerating value within a healthy system.

The sample ROI calculator yields $8,365,000 as the calculated value for the total one-year investment.

> **First year investment = Direct hard costs + J-Curve cost**

1. DORA AI Capabilities Model report, https://dora.dev/dora-aicmr
2. "Capabilities: Continuous integration." https://dora.dev/capabilities/continuous-integration
3. "Capabilities: Test automation." https://dora.dev/capabilities/test-automation
4. "Capabilities: AI-accessible internal data." https://dora.dev/capabilities/ai-accessible-internal-data
5. "DORA's software delivery performance metrics." https://dora.dev/guides/dora-metrics
6. "Capabilities: Working in small batches." https://dora.dev/capabilities/working-in-small-batches
7. "Capabilities: Version control." https://dora.dev/capabilities/version-control

ROI calculation and financial modeling

Translating the benefits of AI into a financial model requires structured and realistic forecasting. The DORA ROI of AI calculator[1] serves as a starting point for this process. This chapter breaks down the financial modeling into three parts: the return on investment calculation, the payback period, and scenario analysis.

For simplicity, the DORA ROI of AI calculator limits its scope to one-year projections. Due to the initial productivity drop associated with the J-Curve, the first year is the most complex period to model accurately. Once you establish a baseline, you can extrapolate these figures to project your returns for the second year and beyond.

Calculate return of investment in the first year

Calculating the return requires separating the financial model into two ledgers: First year return and First year investment. The difference between these two figures represents the First year benefit, which is then divided by the investment to determine the percentage of return.

First year benefit = First year return – First year investment

The sample ROI calculator uses the following estimates:

- Total first year investment: $8.4M
- Total first year return: $11.6M

These estimates yield $3.3M as the calculated value for the First year benefit.

$$\text{First year ROI (\%)} = \frac{\text{First year benefit}}{\text{First year investment}}$$

These estimates in the calculator yield a 39% ROI.

Use DORA's ROI of AI calculator[2] to follow along and project the financial outcomes for your organization.

When executing this calculation, leaders must look beyond the price of the tool itself. Evidence from the Stanford Artificial Intelligence Index shows that raw inference costs for advanced models have plummeted remarkably, dropping by a factor of 280 between November 2022 and October 2024.[3] Because the cost of querying models is approaching zero, the true financial burden of adoption has shifted to governance cost: managing the verification tax, adjusting workflows, and upskilling talent.

Similarly, the value side of the ledger must reflect reality. Research analyzing tens of thousands of engineers reveals that while artificial intelligence yields a 35–40% productivity gain on simple, greenfield tasks, its impact on complex, legacy brownfield code is often 10% or less.[4]

Organizations must plan for a multiyear evolution. The first year establishes a foundational return as teams navigate the initial learning phase and the systemic instability tax.

The second year is when organizations experience a compounding effect. As enterprises transition from simple coding assistants to scaling autonomous agents in production, productivity gains and revenue impacts accelerate exponentially in year two and beyond.

Google Cloud customers found an average of 727% return on their investment in Google Cloud AI in three years.[5]

Project the payback period

The payback period measures the exact amount of time required to recover the initial expenditure and ongoing operational costs. This metric is vital because integration follows the J-Curve trajectory: costs are incurred immediately, while value realization is delayed.

Recent global data from Google Cloud reveals that the average payback period for businesses adopting artificial intelligence tools is about eight months.[6] As a rule of thumb, a highly successful target benchmark for agile teams is a payback period of 6–9 months, while 12–18 months is considered acceptable for larger enterprise rollouts requiring heavy governance.

If your first year benefit is greater than your first year investment, use the standard formula:

$$\text{Payback period} = \frac{\text{First year investment}}{\text{First year return}}$$

In our sample ROI calculator, this gives us a payback period of 0.7 years, which is eight months.

Because the total investment includes the heavy tuition phase of change management, your cumulative return may remain lower than your cumulative investment at the end of the first year. If this occurs, you must roll the unrecovered investment into the following year using the multiyear formula:

$$\text{Payback period} = \text{Completed years} + \frac{\text{Unrecovered investment at start of final year}}{\text{Total return during final year}}$$

By presenting the calculation in this structured format, engineering leaders demonstrate to their leaders that the initial financial deficit is not a failure of the tool but the calculated reality of the adoption lifecycle playing out as planned.

Build scenarios to manage adoption risk

Relying on a single-point estimate for forecasting introduces risk. Planning for different scenarios can increase confidence in the estimates and build trust with finance leaders by acknowledging uncertainty. Exploring a range of outcomes allows you to appropriately size the J-Curve of initial adoption, manage expectations, and identify which underlying systemic investments will yield the highest actual return.

We encourage you to plan for a few scenarios by adjusting estimates or adding multipliers to both expected value and costs.

Adjusted scenario value = Baseline value * Value multiplier

Adjusted scenario cost = Baseline cost * Cost multiplier

- **Conservative scenario:** Apply a value multiplier from 0 to 1 (for example 0.8) to value drivers, such as productivity, revenue, and user experience, to account for slower-than-expected user adoption. For the cost side, apply a cost multiplier higher than 1 (for example 1.5) to costs to account for hidden integration overhead and higher-than-expected code review bottlenecks.
- **Realistic base scenario:** Apply a 1.0 multiplier to both categories. This assumes a standard adoption trajectory.
- **Optimistic scenario:** Apply a value multiplier higher than 1 (for example 1.2) to the value drivers to represent elite team execution and compounded productivity gains. For costs, use a cost multiplier from 0 to 1 (for example 0.8), assuming the existing internal engineering platform is mature enough to absorb the new tools efficiently without generating massive verification overhead.

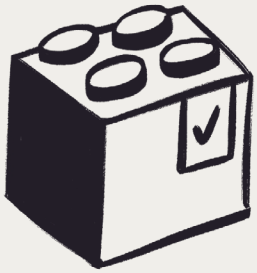
Emphasize the shift in mindset

Simply funding the technology and pushing it into production does not guarantee financial success. Instead, internal organizational capabilities are an earlier predictor of business impact than the financial investments themselves. The financial models outlined in this chapter materialize only when an enterprise possesses the mature structural practices required to govern the technology.

Moving from a theoretical spreadsheet to a realized return requires a fundamental shift in executive focus from the AI tools you purchase to the engineering environment you cultivate.

In the "Build the organizational foundation for AI adoption" chapter, we will explore this transition and define the exact systemic practices that represent the key to adoption.

To explore these calculations further or to seek support from Google Cloud Professional Services, please connect with our team.[7]



1. "ROI of AI calculator." https://dora.dev/ai/roi/calculator
2. "ROI of AI calculator"
3. Stanford HAI, "2025 AI Index." https://hai.stanford.edu/ai-index
4. Stanford University, "Software Engineering Productivity Research." https://softwareengineeringproductivity.stanford.edu
5. Google Cloud, "How Businesses Achieve Strong ROI with Google Cloud AI," 2025, https://cloud.google.com/transform/how-businesses-achieve-strong-roi-with-google-cloud-ai
6. Google Cloud, "How Businesses Achieve Strong ROI"
7. "Contact us." https://dora.dev/ai/roi/contact

delta

This report is a joint effort between Google Cloud's DORA team and Google Cloud's premier innovation practice, delta.

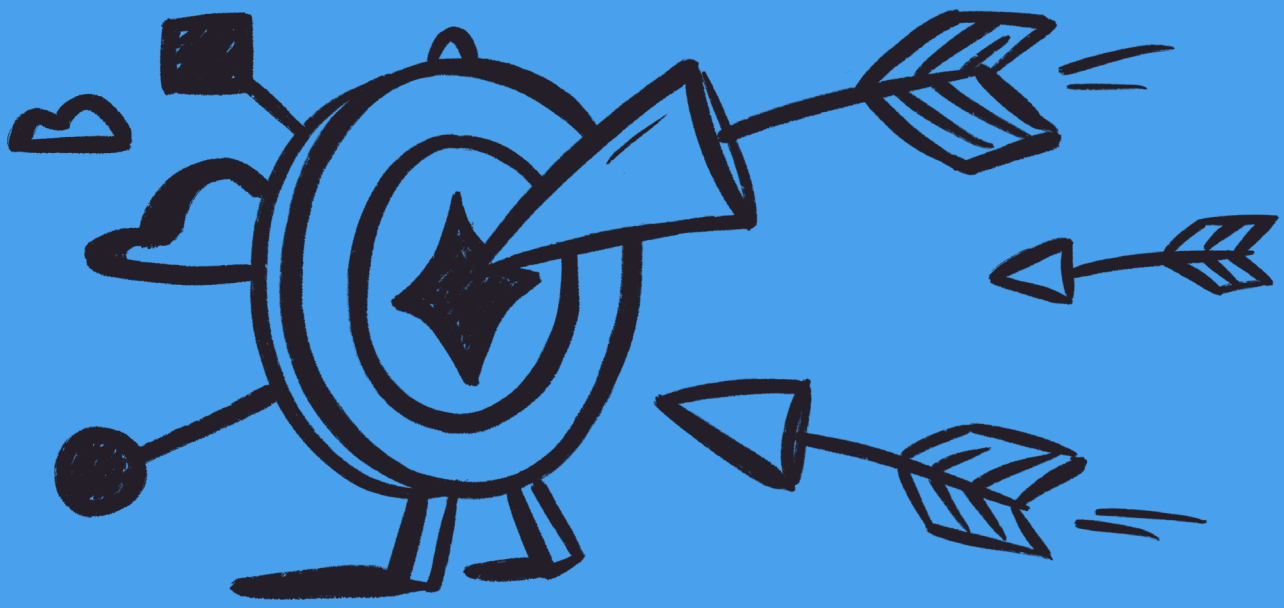
delta is dedicated to bridging the gap between visionary ideas and sustainable business impact, focused on accelerating transformation and fostering diversified growth through a forward-deployed, value-driven model. The multisided approach applies business, technology, and user lenses to every challenge, ensuring that solutions are not only technically scalable but also user-obsessed and operationally efficient.

The delta team is an elite group of Google AI engineers and strategists who specialize in solving the most complex business and technology challenges. A cornerstone of their current mission is agentic transformation, where they enable end-to-end AI-driven transformations—from customer engagement to back-office operations—by building intelligent agent ecosystems that drive measurable P&L impact.

By integrating Google's proven product management, UX, and engineering methodologies with technology and data, delta helps enterprises navigate the entire innovation lifecycle, moving rapidly from initial prototyping to production.

For more information and to see how delta can help you, please visit https://dora.dev/ai/roi/contact.

Build the organizational foundation for AI adoption



Build trust to navigate the J-Curve

Realizing the value of AI in the SDLC is determined not by the sophistication of the large language model (LLM) but by the maturity of the organization. As established in "Understand the market divide on AI returns," AI acts as an amplifier; to ensure it amplifies productivity rather than chaos, organizations must master five systemic keys of adoption.

The most immediate barrier to ROI is the verification tax, the time developers spend reviewing AI outputs. Our research identifies trust in AI as a foundational capability to mitigate this. If trust is low, the J-Curve deepens as developers second guess every line of code, erasing the productivity gains that should be freeing capacity for reinvestment in innovation.

To bridge this, leadership must provide a clear and communicated AI stance.[1] When engineers understand the organizational guardrails and the expectations of learning, they gain the psychological safety required to move from manual skepticism to effective oversight. Trust is not blind. It is a calculated reliance built on a system that rewards verification over raw volume.

Treat your Internal Developer Platform (IDP) as a product

The 2025 research[2] confirms that a quality internal platform[3] is the primary connective tissue for AI value. In the agentic era, an IDP is no longer just a portal for infrastructure, it is the risk mitigator and the context provider for AI agents.

A well-architected, user-centric IDP reduces friction by providing self-service capabilities and standardized workflows. When AI agents can navigate a well-defined platform, they spend less time hallucinating architectural patterns and more time delivering valuable code. The goal is a platform that treats the developer—or any builder—as the user, minimizing the cognitive load required to move code from an agent's prompt to a production environment. Simultaneously, the platform provides the guardrails for the agents who are also users of the platform.

Cultivate AI-accessible data ecosystems

AI agents are only as effective as the data they can access. To move from generic assistance to business-specific value, organizations must invest in AI-accessible internal data.[4] This includes high-quality documentation,[5] clean APIs, and a healthy data ecosystem.[6]

When an organization's internal knowledge graph is fragmented or outdated, AI generates bloat, duplicated or irrelevant code that creates long-term technical debt. Successful adoption requires treating internal technical data as a first-class citizen, ensuring that agents have the ground truth needed to make accurate decisions within the specific context of your code base.

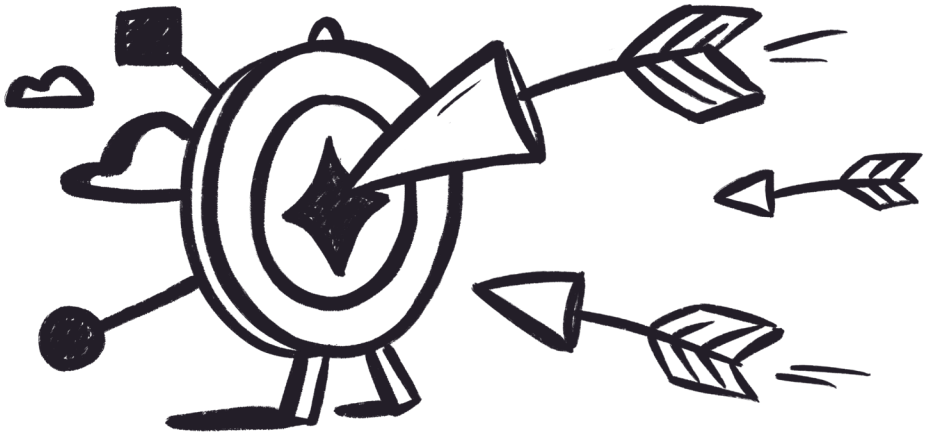
Anchor AI velocity in user value

Finally, adoption must be anchored in the ultimate goal: user value. A user-centric focus[7] ensures that the increased velocity provided by AI is directed at solving real user problems rather than just increasing the frequency of code commits or pull requests.

Deploy guardrails to automate verification

To sustain this velocity without increasing risk, organizations must deploy robust technical guardrails. This involves shifting from manual checkpoints to automated nonoptional security and quality gates. In an agentic world, these guardrails act as the brakes that allow the engineering engine to go faster safely. By automating the verification of AI-generated commits, teams ensure that the increased throughput enabled by the agentic era results in high-quality product, not a higher volume of incidents.

By aligning these five keys—trust, platform, data, users, and guardrails—organizations transform AI from a localized experiment into a systemic engine for financial return.



1. "Capabilities: Clear and communicated AI stance." https://dora.dev/capabilities/clear-and-communicated-ai-stance
2. DORA, *2025 State of AI-assisted Software Development* (2025), https://dora.dev/dora-report-2025
3. "Capabilities: Platform engineering." https://dora.dev/capabilities/platform-engineering
4. "Capabilities: AI-accessible internal data." https://dora.dev/capabilities/ai-accessible-internal-data
5. "Capabilities: Documentation quality." https://dora.dev/capabilities/documentation-quality
6. "Capabilities: Healthy data ecosystems." https://dora.dev/capabilities/healthy-data-ecosystems
7. "Capabilities: User-centric focus." https://dora.dev/capabilities/user-centric-focus

Map your AI investment roadmap

The path to ROI is a sequence of capability building, not a race to adopt the latest model or the latest tool. To ensure the transition to an agentic workflow yields sustainable value, organizations must prioritize the development of the core DORA capabilities.[1] While performance metrics serve as the essential gauge of progress, it is the underlying capabilities that drive the engine.

Build the context layer (CapEx)

Primary capabilities: quality internal developer platforms[2] and healthy data ecosystems.[3]

The first investment must be in the environment in which AI operates. In an agentic world, garbage in, garbage out refers to the context provided to the agent. Remember, AI is an amplifier. Without a robust foundation, AI generates bloat, redundant or low-quality code that creates a long-term maintenance tax.

The investment: Capital should be directed toward developing a quality internal platform and a healthy data ecosystem. This involves centralizing architectural standards and ensuring documentation[4] is high fidelity and machine readable.

The goal: Minimize friction by ensuring agents have a clear, standardized map of the organization's technical landscape.

Empower the human in the loop (OpEx)

Primary capabilities: trust in AI[5] and context engineering.[6]

To navigate the J-Curve productivity dip, the investment must shift from the platform to the person. The cost of adoption is highest when developers struggle to verify agentic output, leading to a high verification tax.

The investment: Organizational expenditures should focus on training for context engineering and verification. Developers must be equipped to act as high-level orchestrators, providing agents with precise business context and maintaining rigorous oversight.

The goal: Build trust in AI through competency. By improving the team's ability to guide and verify agents, organizations reduce the risk of production incidents and shorten the duration of the initial learning curve.

Validate progress with leading indicators

The metrics gauge: experiment frequency and software delivery performance.

With the foundational capabilities in place, organizations can use DORA's software delivery performance metrics[7] as the dashboard to confirm they are heading in the right direction. Before looking for lagging financial returns such as revenue, leaders must monitor the pulse of the engineering system.

Leading metrics: Monitor experiment frequency, which is how often teams leverage agents to explore new solutions, and deployment frequency. These act as the early indicators that the system is successfully absorbing the new technology.

Stability gauge: Track change failure rate and rework to ensure that increased velocity isn't coming at the cost of stability.

By treating capabilities as the primary investment and metrics as the secondary gauge, engineering leaders ensure that their ROI is built on a stable, scalable organizational system rather than a series of isolated, fragile gains.

The financial value of experiment frequency

A critical reality in software development is that most well-designed features and new ideas fail to improve key business metrics. Because the failure rate of even the most promising assumptions is inherently high, betting massive, upfront investments on unverified ideas is a fragile strategy. This is where artificial intelligence acts as a profound value multiplier.

By significantly reducing the time and effort required to write code, gen AI allows teams to quickly build multiple variations of a feature. Instead of converging prematurely on a single, untested idea, teams can experiment with a wider array of options to production. This rapid iteration fundamentally increases a team's user-centric focus. It shifts the development process from internal guessing to external validation, allowing actual user behavior to dictate the right thing to ship.

In financial terms, this strategy leverages the powerful concept of **optionality**.

- **Lowering the option premium:** In finance, an option is a low-risk investment that provides the right—but not the obligation—to make a larger investment later. In software development, every experiment or prototype is an option. AI drastically reduces the upfront "premium" (the time and labor cost) of creating these software options.
- **Maximizing option value:** By increasing experiment frequency, teams build a diverse portfolio of low-cost options. The organization "exercises" the option—committing the expensive, long-term resources required to maintain, scale, and secure the code—only when the experiment explicitly proves it delivers tangible business value.

Therefore, experiment frequency—monitoring how often teams leverage agents to explore new solutions—is not just an engineering measure, it is a leading financial indicator. A high experiment frequency proves that the engineering system is successfully absorbing the new technology and actively minimizing the risk of investing in the wrong features.

1. "Capability catalog." https://dora.dev/capabilities
2. "Capabilities: Platform engineering." https://dora.dev/capabilities/platform-engineering
3. "Capabilities: Healthy data ecosystems." https://dora.dev/capabilities/healthy-data-ecosystems
4. "Capabilities: Documentation quality." https://dora.dev/capabilities/documentation-quality
5. "Fostering developers' trust in generative artificial intelligence." https://dora.dev/insights/trust-in-ai
6. "Capabilities: AI-accessible internal data." https://dora.dev/capabilities/ai-accessible-internal-data
7. "DORA's software delivery performance metrics." https://dora.dev/guides/dora-metrics
8. N. Larsen et al., "Statistical Challenges in Online Controlled Experiments: A Review of A/B Testing Methodology," arXiv (2022), https://doi.org/10.48550/arxiv.2212.11366

Secure long-term ROI

With a clear roadmap in place, it is critical to remember that AI is not a plug-and-play solution—it is a powerful amplifier of your current engineering culture. The organizations that will secure the highest, most sustainable returns are those that actively prepare their systems for this amplification. ROI is measured by how much latent human creativity your organization can unlock through clearing systemic toil. Your ultimate goal is not just to generate code faster but to ensure your underlying platforms and workflows are healthy enough to scale that throughput without magnifying legacy chaos.

Survive the J-Curve of adoption

To realize ROI, organizations must invest in the underlying engineering systems to survive the J-Curve of value realization and turn initial instability into long-term value. Gen AI dramatically increases the volume of code generated, but without a strategic focus on the organizational system, it creates localized pockets of productivity that are often lost to downstream chaos. When organizations deploy AI tools, they should expect a productivity dip during the initial phases of adoption. This dip represents the tuition cost of transformation, driven by workflow friction and a necessary verification tax—the time developers spend reviewing AI outputs. This temporary instability isn't a signal of failure but a necessary investment in learning and adaptation.

Reinvest capacity to drive innovation

As your teams navigate this volatility and unlock new velocity, we strongly recommend organizations do not adopt a headcount-reduction strategy. Instead, productivity gains should be framed as freeing capacity for innovation and value creation. Reducing the amount of unnecessary rework recovers engineering capacity equivalent to "free headcount," which can be directly reinvested into building new features. Retaining and training existing talent is more cost-effective, preserves institutional knowledge, and gives organizations a distinct advantage by having a strong technical workforce that is engaged and continuing to learn.

Translate engineering metrics into financial impact

Building on the leading indicators established in your roadmap, leaders must measure software delivery performance to translate those engineering metrics into financial outcomes. We recommend using DORA's software delivery metrics[1] to evaluate the throughput and instability of software changes. Increasing throughput reduces time-to-market, allowing the business to realize value and recognize returns on new features sooner. Conversely, high levels of instability create operational friction and disrupt revenue streams.

Build a stable foundation for long-term ROI

The path to ROI is a sequence of capability building, not a race to adopt the latest model or the latest tool. Ultimately, the true risk for technology leaders is not failing at AI but failing to build the organizational system and culture that allow AI to deliver value. The cost of inaction is steep, but rushing into adoption without preparing your environment is equally detrimental.

Executing the investment roadmap requires a sustained commitment to the context layer, directing capital toward developing quality internal developer platforms[2] and healthy data ecosystems.[3]

By treating these core capabilities as the primary investment and metrics as the secondary gauge, leaders ensure their ROI is built on a stable, scalable foundation. Invest in engineering excellence today to ensure AI acts as an amplifier of value rather than a catalyst for downstream chaos.

You can read more about these technical capabilities and access comprehensive research data at DORA.dev. Share success stories and continue your learning journey with peers at DORA.community.

1. "DORA's software delivery performance metrics." https://dora.dev/guides/dora-metrics
2. "Capabilities: Platform engineering." https://dora.dev/capabilities/platform-engineering
3. "Capabilities: Healthy data ecosystems." https://dora.dev/capabilities/healthy-data-ecosystems

Acknowledgments

The ROI of AI-assisted Software Development report is a collaboration between DORA[1] and Google Cloud's delta team.[2]

Advisors and contributors

- Ben Jose
- Eric Lam
- Matt Orr
- Allison Park
- Ryan J. Salva
- Jerome Simms
- Dave Stanke
- Cedric Yao

Design partners

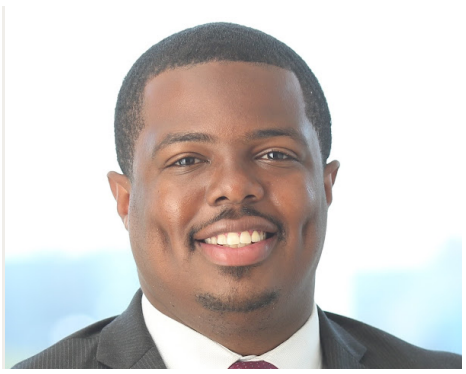
Report layout and design by Human After All (https://www.humanafterall.studio)

Authors

Eva Dong

Eva Dong leads AI Value Realization in the Americas at Google Cloud. With a focus on moving from pilot to impact, she helps organizations navigate the complexities of AI adoption to drive financial ROI.

A data scientist at heart, Eva thrives on turning complex data into business impact. Before Google Cloud, Eva spent eight years at McKinsey & Co in AI and Data Science, leading analytical transformations across the Americas, Europe, Africa, and Asia. She also brings deep technical experience from her time as a Data Scientist at Visa and holds a Master's in Financial Engineering from the University of Michigan, Ann Arbor.



Andre Ellis Jr.

Andre Ellis Jr. is a Cloud Financial Operations Lead who helps clients maximize their savings on Google Cloud and drive financial resiliency for their business.

Andre has deep expertise in technology strategy and finance, and has spent more than a decade helping large enterprises capitalize on their technology investment. Andre has led CXOs on high-impacting engagements to accelerate cost savings and define strategic roadmaps to grow their business. Andre holds a bachelor's degree in Computer Science from Morehouse College and an MBA from the Wharton School of Business.

Nathen Harvey

Nathen Harvey leads the DORA team at Google Cloud. He leverages industry-shaping research to drive product strategy and help organizations improve software delivery speed, stability, and the developer experience. A frequent speaker on DevOps and AI, Nathen is dedicated to building solutions that empower technical communities. He has co-authored multiple DORA reports and contributed to *97 Things Every Cloud Engineer Should Know*, published by O'Reilly in 2020.

Vivian Hu

Vivian Hu is a 10X Technology Consultant at Google Cloud. She is a contributor to DORA's 2025 State of AI-assisted software development report and the DORA AI Capabilities Model report. She works with top global organizations on the intersection of AI and software development excellence and her work helps leaders harness the power of cutting-edge AI systems while grounding them in the foundational principles that ensure stability, user-centricity, and real business value.

Ursula Löbbert-Passing, Ph.D.

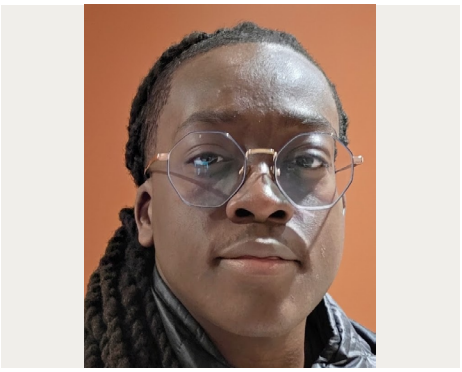
Ursula Löbbert-Passing, Ph.D. is the AI Value Realization Lead in Google Cloud's Professional Services in EMEA. She works with Google Cloud's most strategic customers to help them measure and realize business value from AI solutions.

Ursula brings more than 20 years of experience in benchmarking, strategy, and value advisory. She holds a PhD on effort estimation for software projects—"quantifying the unquantifiable" has been her passion ever since.

Eric Maxwell

With a career that began in the trenches of software engineering, Eric Maxwell has always been driven by a passion for automation and a strong empathy for fellow practitioners. This foundational experience is the cornerstone of his work today as the leader of Google's 10X Technology consulting practice.

Eric advises top global companies on how to deliver value faster by transforming their organizations through strategic improvements in technology, process, and culture. He contributes his expertise to the renowned DORA team. Before his time at Google, Eric was at Chef Software, where he honed his skills in infrastructure automation and whipping up awesome.



Aaron Wanjala

Aaron Wanjala is a Spring Boot and Angular developer whose career has taken him from startups to large corporate environments and back again. As a developer, he's had the opportunity to build and ship applications in sectors including financial services and identity management.

As a cloud developer advocate, he's excited about making the process easier for developers at organizations of all sizes, with a particular focus on the intersection between app modernization and AI code assistance. Most importantly, Aaron loves telling stories based on his recent experience and sharing his excitement about new technologies with other developers.

1. "DORA." https://dora.dev
2. "Innovation and transformation services." https://cloud.google.com/consulting/innovation-and-transformation

Next steps

Join the DORA Community to discuss, learn, and collaborate on improving the impact of technology-driven teams and organizations. https://dora.community

DORA ROI of AI-assisted software development calculator

An interactive version of the ROI calculator is available at https://dora.dev/ai/roi/calculator.

Contact us

Google Cloud Consulting's delta team of experts and innovators is ready to help you continue the conversation and build a strong value model. Contact us today.[10]

Check if you have the latest version of this report

Visit https://dora.dev/vc/airoi/?v=2026.1 to see if there's a newer version of this report available.

Unless otherwise noted, all citations retrieved February, 2026.

Recommended reading

More from DORA

2025 State of AI-assisted Software Development[1]

DORA AI Capabilities Model[2]

The ROI of DevOps Transformation[3]

Explore DORA AI-related research, findings, and recommendations. https://dora.dev/ai

Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution. https://itrevolution.com/product/accelerate

More from Google Cloud

The ROI of AI 2025: How agents are unlocking the next wave of AI-driven business value[4]

How businesses achieve strong ROI with Google Cloud AI[5]

Three-part framework to measure the impact of your AI use case[6]

Key insights from our inaugural survey on the ROI of AI in the public sector[7]

How to calculate your AI costs on Google Cloud[8]

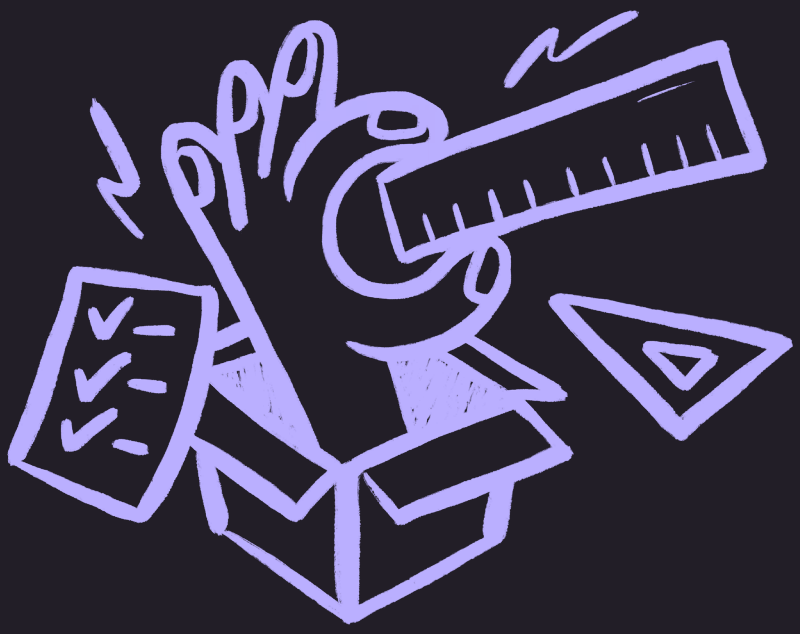
Leveraging Cloud FinOps to measure the business value realized by your cloud transformation[9]

1. DORA, *2025 State of AI-assisted Software Development* (2025), https://dora.dev/dora-report-2025
2. DORA, *DORA AI Capabilities Model* (2025), https://dora.dev/dora-aicmr
3. DORA, *The ROI of DevOps Transformation* (2020), https://dora.dev/dora-report-2020
4. "The ROI of AI 2025: How agents are unlocking the next wave of AI-driven business value." https://cloud.google.com/resources/content/roi-of-ai-2025
5. "How businesses achieve strong ROI with Google Cloud AI." https://cloud.google.com/transform/how-businesses-achieve-strong-roi-with-google-cloud-ai
6. "Three-part framework to measure the impact of your AI use case." https://cloud.google.com/blog/topics/cost-management/measure-the-value-and-impact-of-your-ai
7. "Key insights from our inaugural survey on the ROI of AI in the public sector." https://cloud.google.com/blog/topics/public-sector/key-insights-from-our-inaugural-survey-on-the-roi-of-ai-in-the-public-sector
8. "How to calculate your AI costs on Google Cloud." https://cloud.google.com/blog/topics/cost-management/unlock-the-true-cost-of-enterprise-ai-on-google-cloud
9. "Leveraging Cloud FinOps to measure the business value realized by your cloud transformation." https://services.google.com/fh/files/misc/leveraging_finops_to_measure_the_business_value_realised_by_the_cloud_transformation.pdf
10. "Contact us." https://dora.dev/ai/roi/contact

Appendix: Sample ROI calculator

This is a sample ROI calculator that is meant for demonstration purposes only. Treat these calculations as a high-uncertainty estimate meant to spark a conversation rather than a rigid mathematical formula. As the saying goes, all models are wrong,[1] but we hope this one proves useful for your team as you assess the impact of AI in the software development lifecycle.

As with any DORA insights, the real value comes from contextualizing the findings and applying them within your own organization.



Input variables

Organizational metrics

Technical staff size in number of full-time employees (FTE): The count of full-time employees, or equivalent, who are involved in the software development lifecycle. Covers all roles that may use AI to contribute to the process, including product managers, software engineers, user experience designers, technical leads, site reliability engineers, and more. The sample calculator uses an estimate of 500 FTEs.

Average fully loaded technical staff salary: Software engineer salaries vary greatly between regions in the world. The "fully loaded" cost (total cost to the employer) typically adds between 30% (U.S.) and 100% (Europe) on top of the average base salary to account for taxes, benefits, and overhead. The sample calculator uses a blended rate of $176,000.

Baseline software delivery metrics

Product portfolio revenue: The annual revenue driven by this software. The sample calculator uses an estimate of $100,000,000.

Cost of downtime per hour: An estimate of the cost of one hour of system outage. We recommend considering revenue lost as well as additional costs, such as reputational damage. You could also estimate this by dividing the product portfolio revenue by the number of hours in a year (8,760), though this has a lot of assumptions built in. The sample calculator uses an estimated cost of $100,000 per hour.

Current number of deployments per year: The total number of deployments per year for this application or service. The sample calculator uses an estimate of 50 deployments per year, about once per week.

Current number of features deployed per year: The number of features deployed each year. Features are user-facing enhancements designed to improve the product experience. Features and deployments rarely have a one-to-one relationship; a single deployment might bundle several new features, or a single complex feature might be rolled out incrementally across multiple deployments. The sample calculator uses the deployments per year as a reasonable proxy for this and estimates 50 features per year, about one per week.

Idea success rate: The percentage of deployed features that increase revenue. The sample calculator estimates that about one-third (33%) of features shipped increase revenue.

Average revenue impact per successful feature: The average revenue increase for each successful feature. We recognize this is very difficult to project and recommend a conservative value here between 0.01% and 1%. The sample calculator uses 0.5%, a value in the middle of that range. Given a product portfolio revenue of $100,000,000, this calculates to $500,000 per successful feature.

Current change failure rate (CFR): The current percentage of changes to production or released to users that results in degraded service. The sample calculator uses 5%.

Failed deployment recovery time (FDRT): The average number of hours it generally takes to restore service after a change to production or release to users results in degraded service. The sample calculator uses four hours.

AI estimates

Net time saved per developer: The net productivity boost per developer, captured as a percentage. Be sure to consider both time saved in generating solutions and the verification tax associated with things like code reviews. There are many sources for time-saving estimates, and the estimates vary between 40 minutes and 150 minutes per day in software development. The sample calculator uses a conservative estimate of 12.5% or about one hour of an eight-hour working day.

Annual AI license cost per user: The annual price per user of an AI subscription. The sample calculator uses an estimate of $250, slightly higher than $20 per month.

Additional annual AI costs per user: The additional per-user costs, such as API or token costs. The sample calculator uses $80 per year.

Additional annual AI infrastructure costs: The new AI-related infrastructure costs including compute, networking, storage, and monitoring. The sample calculator uses $100,000.

Annual training costs per user: The training and enablement costs for each employee. The sample calculator uses $9,600.

Target number of deployments per year: The number of expected deployments per year while using AI in the SDLC. DORA's research shows that adopting AI is associated with an increase in software delivery throughput.[2] The sample calculator estimates this as 56 deployments per year, a 12% increase over the baseline number of deployments.

Target number of features deployed per year: The number of expected features deployed per year. Features are user-facing enhancements designed to improve the product experience. Features and deployments rarely have a one-to-one relationship; a single deployment might bundle several new features, or a single complex feature might be rolled out incrementally across multiple deployments. The sample calculator uses the target deployments per year as a reasonable proxy for this and estimates 56 features per year, a 12% increase over the baseline number of features deployed.

Target change failure rate (CFR): The target percentage of changes to production or released to users that result in degraded service. DORA's research shows an increase in delivery instability is associated with the adoption of AI.[3] The sample calculator uses 6%, a 20% increase over the baseline of 5%.

J-Curve productivity drop: The temporary productivity decrease during the AI learning phase. The sample calculator estimates this at 15%.

J-Curve productivity drop timeline: The length of the productivity decrease. The sample calculator estimates this at three months.

Calculated variables

Costs

Total hard costs (tooling and training): This calculation multiplies the cost of licenses, additional AI costs, and training costs by the number of FTEs and then adds on the additional infrastructure costs. The sample calculation gives $5,065,000.

J-Curve cost: This multiplies the number of FTE, the fully loaded salary, the J-Curve productivity drop, and the J-Curve duration to estimate the cost of the J-Curve. The sample calculation gives $3,300,000.

Total first year investment: This is the sum of the total hard costs and the J-Curve cost. The sample calculation gives $8,365,000.

Value

Headcount reinvestment capacity: An estimate of the capacity that has been freed up for activities like innovation and value creation. This may also represent the avoided costs of additional hiring. This is calculated by multiplying the number of FTE, the fully loaded salary, and the time saved per developer. Productivity gains from net time saved free up capacity for innovation and value creation, and rule out the need to hire additional staff. The sample calculation gives $11,000,000.

Revenue from extra feature deployments: An estimate of the incremental revenue earned by the change in the number of successful features deployed. The sample calculation gives $990,000.

Downtime impact: An estimate of the "instability tax" calculated by looking at the changes in both deployment frequency and the change failure rate after AI is incorporated into the software delivery process. The sample calculation gives –$344.000.

The sample calculator shows an increase in software delivery instability which is not overcome by the incremental value of successful features deployed. Note that the sample calculator keeps the failed deployment recovery time constant at four hours.

Total annual value: This is the sum of the headcount reinvestment capacity, the revenue from new features, and the change in cost of downtime. The sample calculation gives $11,646,000.

Summary

First year benefit: The difference between the annual return and the first year investment. The sample calculation gives $3,281,000.

Return on investment (ROI): The first year benefit divided by the first year investment. The sample calculation gives 39%.

Payback period (years): The first year investment divided by the annual return. The sample calculation gives 0.7 years, or about eight months.

Sample calculations

This is a sample ROI calculator that is meant for demonstration purposes only.

Organizational metrics	
Technical staff size in number of full-time employees (FTE)	500
Average fully loaded technical staff salary	$176,000

Baseline software delivery metrics	
Product portfolio revenue	$100,000,000
Cost of downtime per hour	$100,000
Current number of deployments per year	50
Current number of features deployed per year	50
Idea success rate	33%
Average revenue impact per successful feature	0.5%
Current change failure rate (CFR)	5%
Failed deployment recovery time (FDRT)	4 hours

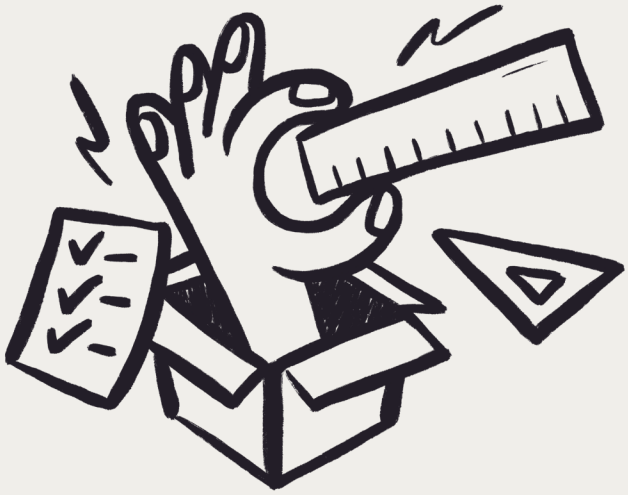
AI estimates	
Net time saved per developer	12.5%
Annual AI license cost per user	$250
Additional annual AI costs per user	$80
Additional annual AI infrastructure costs	$100,000
Annual training costs per user	$9,600
Target number of deployments per year	56
Target number of features deployed per year	56
Target change failure rate (CFR)	6%
J-Curve productivity drop	15%
J-Curve productivity drop timeline	3 months

Costs	
Total hard costs (tooling and training)	$5,065,000
$5,065,000	$3,300,000
J-Curve cost	$8,365,000

Value	
Headcount reinvestment capacity	$11,000,000
Revenue from extra feature deployments	$990,000
Downtime impact	-$344,000
Total first year value	$11,646,000

Summary	
First year benefit	$3,281,000
Return on investment (ROI)	39%
Payback period (years)	0.7 years

An interactive ROI of AI-assisted software development calculator[4] is available so you can explore the mechanics, adjust the assumptions to match your reality, and build your own estimate. Once you've had a chance to experiment, we invite you to share how it's working out for your team in the DORA Community.[5]



1. "All models are wrong." https://en.wikipedia.org/wiki/All_models_are_wrong
2. DORA, *2025 State of AI-assisted Software Development* (2025), https://dora.dev/dora-report-2025
3. DORA, *2025 State of AI-assisted Software Development*
4. "DORA ROI of AI-assisted software development calculator." https://dora.dev/ai/roi/calculator
5. "DORA Community." https://dora.community

Get better at getting better

dora.dev