Gartner

Gartner Insights

2026 Planning Guide for Software Engineering

Matt Brasier, Bill Holz, Roy Triesscheijn, Johnny Walters, Sushant Singhal, Erin Khoo, Danny Brian, Paige Kirk

2026 Planning Guide for Software Engineering

13 October 2025 - ID G00837698 - 42 min read

By: Matt Brasier, Bill Holz, Roy Triesscheijn, Johnny Walters, Sushant Singhal, Erin Khoo, Danny Brian, Paige Kirk

Initiatives: Software Engineering for Technical Professionals; Generative AI Resource Center

In 2026, software engineering teams will be dealing with demands for increased productivity, more use of AI and improved security. Technical professionals should focus on software engineering fundamentals, implement platform engineering and build bridges with security to navigate this environment.

Overview

Key Findings

- Organizations are shifting focus away from developer experience and toward developer productivity. However, most organizations do not have a clear definition of developer productivity or an established baseline for it.

- Platform engineering continues to be a focus for streamlining developer work, but building developer platforms without treating developers as customers will fail to deliver the benefits they anticipate.

- Both the frequency and impact of security incidents continue to increase, yet many organizations still perform security theatre without achieving real-world improvements.

- AI tools offer useful, though not world-changing, benefits to software engineers. However, many organizations hit barriers to adoption or fail to realize the full benefits of these tools due to unrealistic management expectations and a misunderstanding of what the job of the software engineer entails.

Recommendations

- Improve developer productivity by focusing on software engineering basics, such as product-centric delivery, so that when you introduce AI tools and emerging approaches, you maximize their benefits.

- Reduce developer overload by using platform engineering to simplify nondifferentiating work and enhance developer knowledge.

- Break the cycle of software engineers and security professionals seeing each other as the problem by adopting the role of security champion, proactively reaching out and building bridges with software security teams.

- Capitalize on the value of AI tools by building on developers' natural curiosity and desire to learn. Use AI for tasks across the software development life cycle (SDLC) where the work does not require a strong understanding of your organization and its goals.

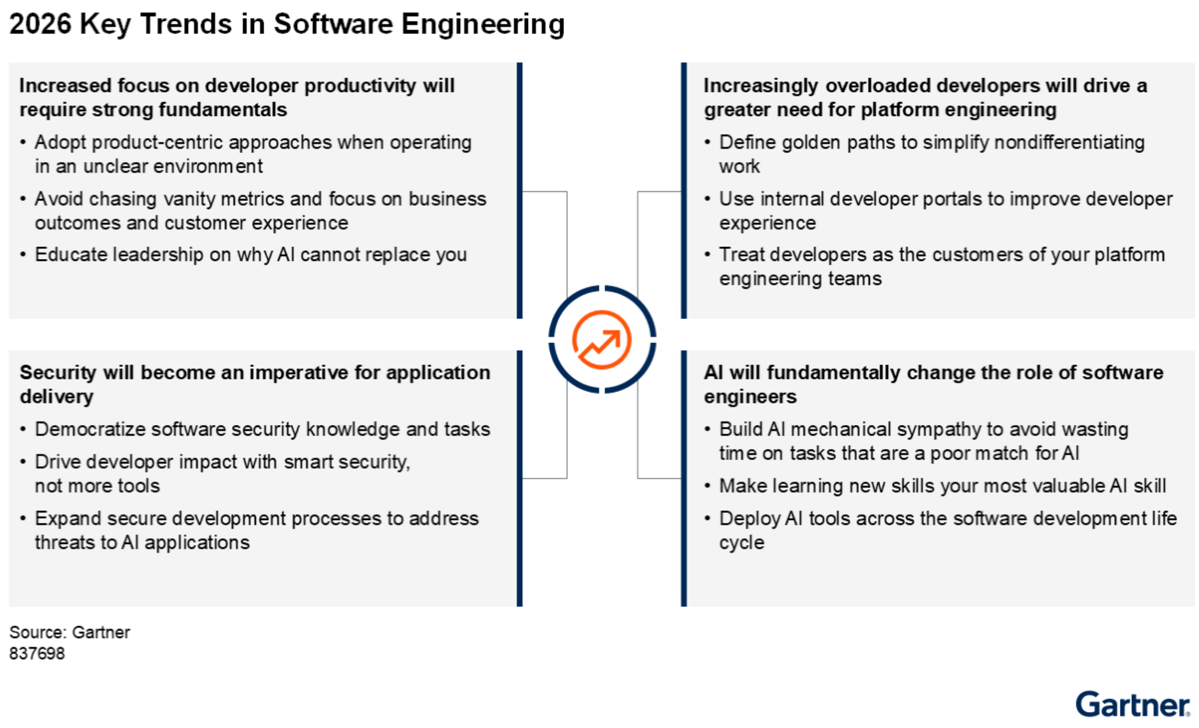
Software Engineering Trends

2026 will be another year in which software engineering teams are pulled in multiple directions. Requests to use AI to go faster, while at the same time being asked to justify its use, are contradictory yet commonplace. Because software engineering has emerged as one of the disciplines where generative AI (GenAI) can offer concrete — if not game-changing — value, software engineering teams are experiencing increased scrutiny at the board level.

To thrive in 2026, software engineering teams must continue to sharpen and shore up the skills that are key to operating in a fast-changing environment, while building solid foundations for delivering secure applications that delight customers.

Figure 1 outlines Gartner's four leading trends that will impact software engineers in 2026, along with their associated planning considerations. These trends have been compiled across Gartner's team of technical software engineering analysts, drawing on client inquiries, conference and survey data, and hands-on research.

Figure 1: 2026 Key Trends in Software Engineering



2026 Key Trends in Software Engineering

Increased focus on developer productivity will require strong fundamentals

- Adopt product-centric approaches when operating in an unclear environment
- Avoid chasing vanity metrics and focus on business outcomes and customer experience
- Educate leadership on why AI cannot replace you

Increasingly overloaded developers will drive a greater need for platform engineering

- Define golden paths to simplify nondifferentiating work
- Use internal developer portals to improve developer experience
- Treat developers as the customers of your platform engineering teams

Security will become an imperative for application delivery

- Democratize software security knowledge and tasks
- Drive developer impact with smart security, not more tools
- Expand secure development processes to address threats to AI applications

AI will fundamentally change the role of software engineers

- Build AI mechanical sympathy to avoid wasting time on tasks that are a poor match for AI
- Make learning new skills your most valuable AI skill
- Deploy AI tools across the software development life cycle

Source: Gartner
837698

Gartner.

Gartner's four key trends for software engineering in 2026 are (click links to jump to sections):

- Increased focus on developer productivity will require strong fundamentals.
- Increasingly overloaded developers will drive greater need for platform engineering.
- Security will become an imperative for application delivery.
- AI will fundamentally change the role of software engineers.

Increased Focus on Developer Productivity Will Require Strong Fundamentals

Back to top

If there's one thing that remains constant in software engineering, it's that everything is always changing — whether that change comes from new architectures or frameworks, new techniques or new delivery models. As a result, the fundamental best practices of software engineering have evolved to focus on enabling change rather than fighting it.

Software engineers should follow these planning considerations as they strengthen fundamental processes and define and increase delivery:

- Adopt product-centric approaches when operating in an unclearenvironment.
- Avoid chasing vanity metrics and focus on business outcomes and customer experience.
- Educate leadership as to why AI cannot replace you.

Planning Considerations

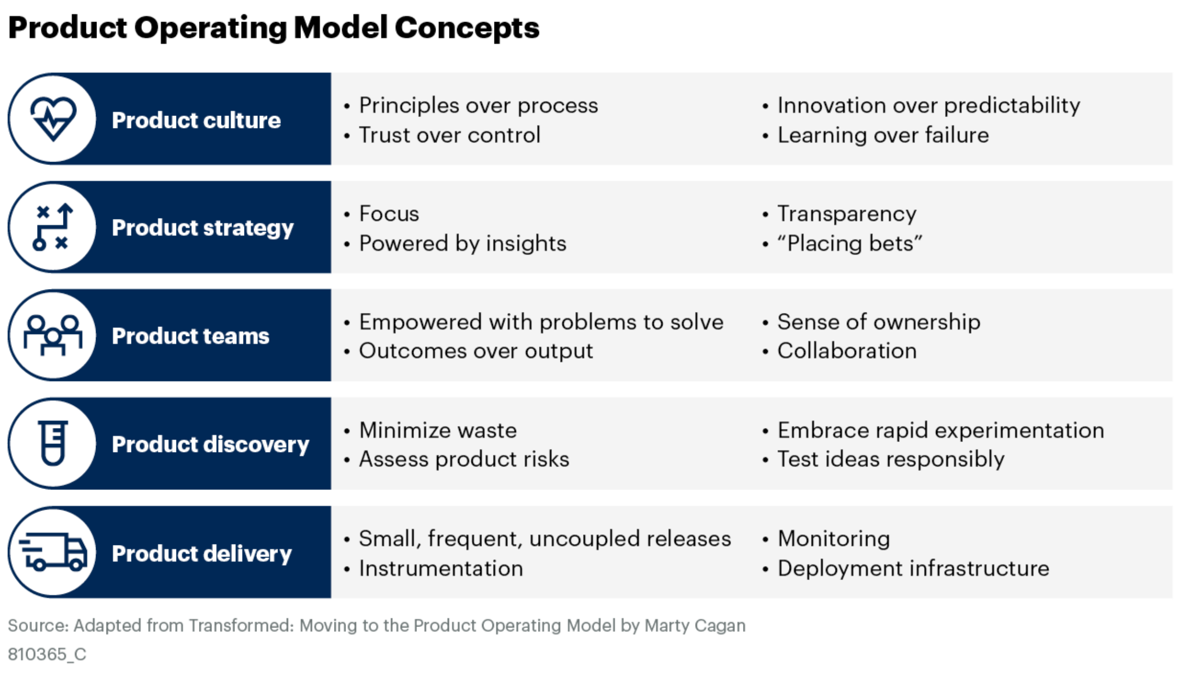
Adopt Product-Centric Approaches When Operating in an Unclear Environment

Back to top | Back to planning considerations list

A product-centric operating model focuses software engineering teams on delivering and maintaining applications over their entire lifetime, whereas a project-centric model views teams as responsible for delivering a specific release of software. Adopting a product-centric model requires teams to develop an ongoing understanding of customers and their needs. Successful software products are not delivered as one-off projects that enter maintenance mode once they go live; instead, they continuously evolve, with regular feedback from customers driving an ever-changing roadmap.

Figure 2 illustrates the core concepts that make up a product-centric development model.

Figure 2: Product Operating Model Concepts



Product Operating Model Concepts

Product culture	• Principles over process • Trust over control	• Innovation over predictability • Learning over failure
Product strategy	• Focus • Powered by insights	• Transparency • “Placing bets”
Product teams	• Empowered with problems to solve • Outcomes over output	• Sense of ownership • Collaboration
Product discovery	• Minimize waste • Assess product risks	• Embrace rapid experimentation • Test ideas responsibly
Product delivery	• Small, frequent, uncoupled releases • Instrumentation	• Monitoring • Deployment infrastructure

Source: Adapted from Transformed: Moving to the Product Operating Model by Marty Cagan
810365_C

While individual software engineers or teams may not have the influence to change the funding model they operate under, they can still adopt key concepts — such as small, frequent releases — and focus on improving their own collaboration, especially with customers.

Conduct customer surveys, listen to suggestions and hold workshops across user groups to identify customer pain points. Combine this human insight with operational data — such as usage logs — to understand software use and how the needs of users are changing. Work with product leadership to build a long-term roadmap that continually improves and updates your software as user needs evolve.

For more information on organizational design in software engineering, see Organize for Agility With Team Topologies.

Avoid Chasing Vanity Metrics and Focus on Business Outcomes and Customer Experience

Back to top | Back to planning considerations list

Chasing high code coverage numbers, focusing on burndown charts or boasting about lines of code written may create the appearance of progress and effectiveness to stakeholders. However, these metrics do not provide insights into the primary objective of product development: delivering business value. These are vanity metrics, or metrics that create a false sense of product success, efficiency and productivity. Writing more software (or more lines of code) faster does not necessarily mean more value. While these metrics might suggest success, they fail to help teams understand how to improve, deliver true customer value and satisfy business goals.

Software engineering teams are under intense pressure to deliver more with less. In this environment, continuously tracking metrics that clearly reflect business impact and user satisfaction is key to ensuring your team is producing value. Metrics should provide insights into areas that need improvement, help build team confidence and better inform product decisions.

> **Ask yourself, "Can this metric lead to a course of action or inform a decision?" If the answer is "no" or "I don't know," then you should probably reevaluate it.**
>
> *— Source: "Vanity Metrics: Definition, How to Identify Them, and Examples," Tableau.*[1]

To avoid selecting metrics that fail to provide valuable insights, use the following guiding principles:

- Measure business outcomes, such as improved page load time, rather than output metrics such as lines of code.
- Make satisfying customer needs and delivering business value your primary measure of success.
- Measure teams rather than individuals to promote team cohesion and shared accountability.
- Never use metrics to reward, shame or compare individuals, teams or products.
- Select metrics that drive desired behaviors and outcomes.

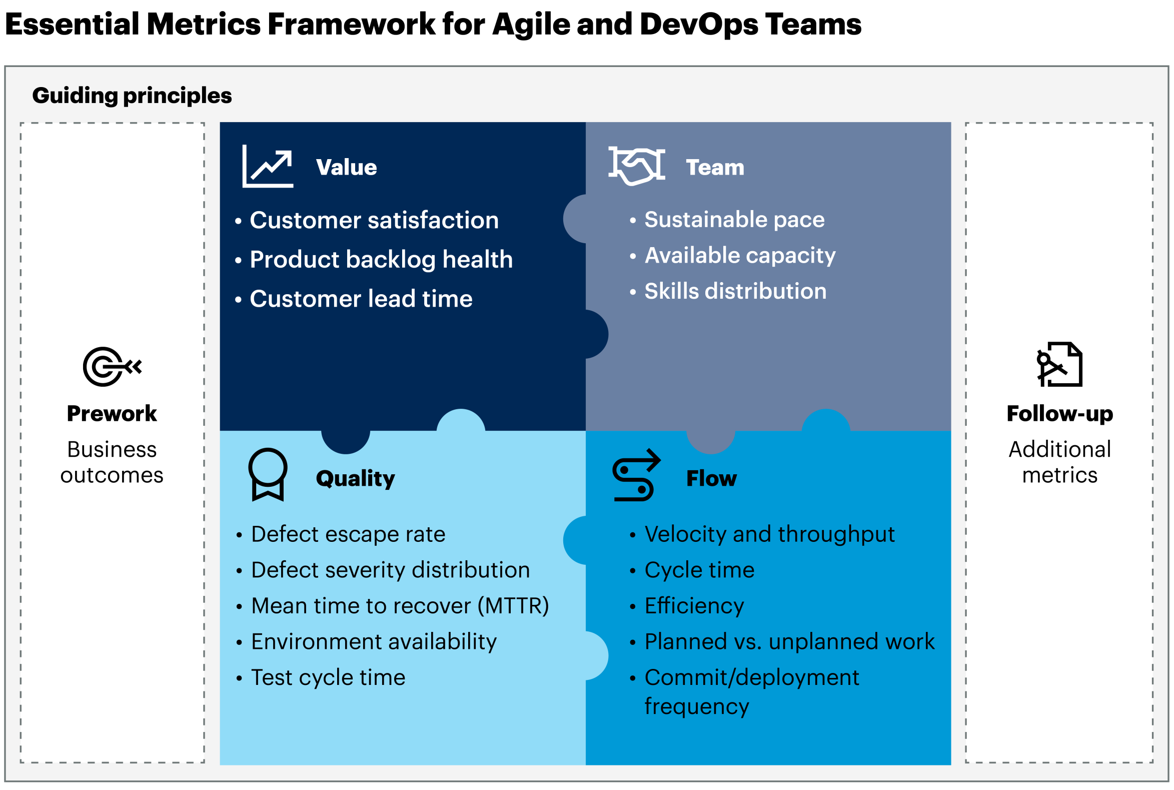
- Use metrics to forecast rather than commit to delivery.
- Make metrics visible, consumable and easily accessible to all team members.
- Use temporary diagnostic metrics to uncover underlying issues.

To deliver great and differentiated products, you first need to identify your goals and understand your customers' needs. Map your business strategy — such as expressed through objectives and key results (OKRs) — onto epics and features, clearly linking contributions to success. This fosters a shared understanding across the team, visualizing how their work directly impacts business outcomes.

Prioritize development that addresses the key pains and gains of your users. Shift to a more outside-in perspective by getting out of the office and observing how your users interact with your product in their real-world environments. This will help you gain insights into their behaviors, needs and goals.

Keeping the guiding principles of value metrics in mind, focus on measuring the four key areas shown in Figure 3: value, team, flow and quality.

Figure 3: Essential Metrics Framework for Agile and DevOps Teams



Source: Gartner
826197_C

Build products with confidence by tracking these key areas:

- **Value metrics** assess whether you are delivering the right product, meeting customer needs and satisfying business goals. If you are not demonstrating value, then all other metrics are irrelevant. Value metrics ensure that your team builds meaningful products that solve customer problems. Additionally, they enable teams to clearly see the impact of their work, giving them a sense of purpose and pride. Adopt value metrics such as customer satisfaction — gathered through customer surveys — and product backlog health. These can be a strong indicator of the state of the product roadmap.

- **Team metrics** measure your overall team's health in areas such as happiness and satisfaction, helping to assess the developer experience. Poor team health can negatively impact talent retention, team velocity and, ultimately, customer satisfaction and value delivery. These metrics are difficult to measure with analytics tools. Instead, use surveys and retrospectives regularly to evaluate team metrics such as sustainable work pace, available capacity and skill distribution. This will improve the developer experience.

- **Flow metrics** evaluate whether your team and delivery processes can identify and meet evolving customer demands and address unanticipated incidents. Failure to prevent, detect and address defects in a timely manner will ultimately have a negative impact on customer satisfaction and retention. Flow metrics provide visualization into the delivery pipeline, enabling your team to identify inefficiencies and create more streamlined processes. Adopt flow metrics such as velocity and throughput, cycle time and efficiency to identify and reduce pipeline bottlenecks.

- **Quality metrics** assess the quality of the end product, as well as its underlying software practices and processes. Poor quality has a domino effect — it inevitably leads to poor flow and lower user satisfaction. Make quality a continuous practice by building it into your development process to avoid an expensive, time-consuming and reputation-damaging fix later. In turn, this frees developers to dedicate more time to adding customer value. Adopt quality metrics such as defect escape rate, mean time to recover and accessibility.

Related Research

- Essential Metrics for Agile and DevOps Teams
- Essential Software Metrics: DORA
- Essential Skills for Software Engineers
- How to Measure and Improve Developer Productivity
- Key Software Engineering Metrics for Testing and Quality
- How to Measure the Value of User Experience Design

Educate Leadership as to Why AI Cannot Replace You

Back to top | Back to planning considerations list

Reducing software engineering headcount due to innovations around AI is premature and might be the wrong decision. While developer productivity gains are real, they're modest and not enough for wholesale reductions. In fact, the gains from AI are likely to be outstripped by ever-increasing demand for more software driving the need for additional developer headcount. More importantly, your future success and impact as a developer depend on creativity, innovation and subtle adaptations that increase value in ways that AI cannot.

AI code assistants typically drive a 10% to 20% increase in developer productivity. Most of the gains come from enhanced velocity and faster code understanding. While these gains are modest, they can actually be transformative to software engineering teams, leading to improved developer satisfaction, increased cognitive bandwidth and automation of rote tasks.

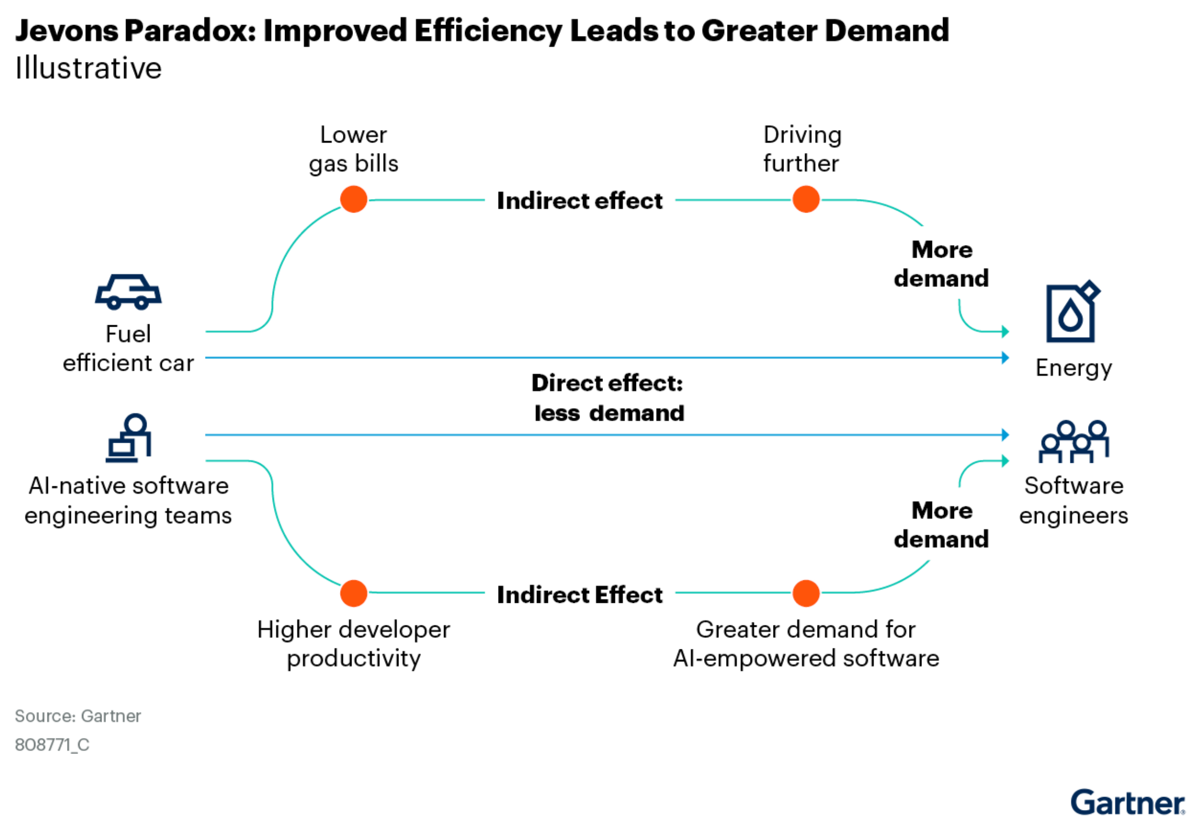
However, these benefits are not enough to drive major reductions in headcount. Differences in perceived productivity gains are due to large software organizations and AI-assisted software tool vendors overstating the extent of end-to-end software development automation. This is further exacerbated by code assistants and agentic IDEs lacking detailed, feature-use analytics, which obscures the true impact of AI-powered software development productivity across the SDLC.

Gartner predicts strong demand for software development across industries in coming years. The enterprise application software market is forecast to grow at a compound annual growth rate (CAGR) of 13.9% through 2028 (see Forecast Analysis: Enterprise Application Software, Worldwide). This growth in software development demand is likely to outstrip efficiencies gained from using AI.

The responsibility for building, integrating and transforming software still lies with software engineers. Their ability to adapt and utilize the full value of technology to drive business outcomes cannot be qualified by productivity metrics alone.

AI will enable software engineers to be more productive, but organizations will actually need more software engineers to meet the increasing demand for software. This phenomenon is known as the Jevons Paradox. [2] As resource efficiency improves, it stimulates demand and expands the scope of resource utilization instead of reducing overall usage. See Figure 4 for a comparison of efficiency improvements with a vehicle.

Figure 4: The Jevons Paradox: Improved Efficiency Leads to Greater Demand


Jevons Paradox: Improved Efficiency Leads to Greater Demand
Illustrative
Lower gas bills
Indirect effect
Driving further
More demand
Fuel efficient car
Energy
Direct effect: less demand
AI-native software engineering teams
Software engineers
More demand
Higher developer productivity
Indirect Effect
Greater demand for AI-empowered software
Source: Gartner
808771_C
Gartner

The time savings from AI adoption in the SDLC are modest, but meaningful; the extra bandwidth is best invested in improving code quality and technical skill development. Read How to Capture AI-Driven Productivity Gains Across the SDLC for more details on the compounding effect of AI productivity.

Although AI is evolving, software engineering and its roles will change in scope. A software engineer is still uniquely positioned to scale enterprise value. More tasks and responsibilities will be augmented with AI, but the best solutions will still require that you put yourself in the user's position and understand their needs and wants. In short, applying human empathy will be a key differentiator as machines continue to accelerate the more repetitive and less differentiated tasks in software development.

In Essential Skills for Building Generative AI Applications and Essential Skills for Quality Engineers, we outline how human evaluation, maintaining high standards of quality, and prioritizing software quality over automation are essential aspects of the SDLC in the AI era.

To fully utilize the benefits of AI, software engineers need to take ownership of the final software product and be involved in ensuring the satisfaction of users. AI systems are not able to define software that is nuanced enough to have full context of your organization's ethos, goals and user needs. Human intuition, empathy and emotional understanding are still critical skills for creating software that delights users.

Related Research

- AI Will Not Replace Software Engineers (and May, in Fact, Require More)
- Innovation Insight for AI-Native Software Engineering
- Predicts 2025: Navigating the Rise of AI in Software Engineering

Increasingly Overloaded Developers Will Drive a Greater Ne ed for Platform Engineering

Back to top

Platform engineering emerged in response to the increasing cognitive load from the complexity of modern software development. Modern software engineering takes place within an increasingly complex technology ecosystem composed of technologies that span the software delivery life cycle — tools, frameworks, continuous integration/continuous delivery (CI/CD) pipelines, testing, security, execution environments, and observability — all enabled by pervasive automation. Developers must often build and operate an assembly of complicated services, such as cloud-based Kubernetes clusters. While having flexibility to solve problems in different ways can be beneficial, when teams needlessly implement different solutions to the same problem, it increases the work required to understand, maintain and govern them.

Follow these planning considerations:

- Define golden paths to simplify nondifferentiating work.
- Use internal developer portals to improve developer experience.
- Treat developers as the customers of your platform engineering teams.

Planning Considerations

Define Golden Paths to Simplify Nondifferentiating Work

To fully utilize the benefits of AI, software engineers need to take ownership of the final software product and be involved in ensuring the satisfaction of users. AI systems are not able to define software that is nuanced enough to have full context of your organization's ethos, goals and user needs. Human intuition, empathy and emotional understanding are still critical skills for creating software that delights users.

Related Research

- AI Will Not Replace Software Engineers (and May, in Fact, Require More)
- Innovation Insight for AI-Native Software Engineering
- Predicts 2025: Navigating the Rise of AI in Software Engineering

Increasingly Overloaded Developers Will Drive a Greater Need for Platform Engineering

Back to top

Platform engineering emerged in response to the increasing cognitive load from the complexity of modern software development. Modern software engineering takes place within an increasingly complex technology ecosystem composed of technologies that span the software delivery life cycle — tools, frameworks, continuous integration/continuous delivery (CI/CD) pipelines, testing, security, execution environments, and observability — all enabled by pervasive automation. Developers must often build and operate an assembly of complicated services, such as cloud-based

Actionable, objective insights

Position your IT organization for success. Explore these additional, complementary resources and tools for software and enterprise application leaders and their teams:

Insights
Key Software Architecture Technologies, Trends and Innovations

Equip your teams with the latest platforms and practices.

Learn More

Insights
Software Engineering for Technical Professionals

Get actionable, objective insights and expert guidance to support mission-critical business initiatives.

Learn More

Webinar
5 Proactive Secure Coding Practices for Software Engineers

Leverage lessons learned through professional experience.

Watch Now

Tool
The Gartner Developer Experience Assessment

Measure, benchmark and improve your team's developer experience.

Learn More

Already a client?
Get access to even more resources in your client portal. Log In

Connect with us

Get actionable, objective business and technology insights that drive smarter decisions and stronger performance on your mission-critical priorities. Contact us to become a client:

U.S.: 1 855 322 5484

International: +44 (0) 3300 296 946

Become a Client

Learn more about Gartner for Technical Professionals
gartner.com/en/information-technology

Stay connected to the latest insights

Attend a Gartner conference
View Conference

Gartner®