Gartner Research

How to Manage Blended Teams of In-House and Outsourced Software Engineering Personnel

Jaideep Thyagarajan, Gunjan Gupta,
Oleksandr Matvitskyy

13 February 2023

Gartner

How to Manage Blended Teams of In-House and Outsourced Software Engineering Personnel

Published 13 February 2023 - ID G00781066 - 15 min read

By Analyst(s): Jaideep Thyagarajan, Gunjan Gupta, Oleksandr Matvitskyy

Initiatives: Software Engineering Leadership; Build a World-Class Software Engineering Organization

Reliance on outsourced software engineering personnel is on the rise, making effective collaboration between in-house and external teams a priority. Software engineering leaders should use Gartner's five principles to manage blended teams and achieve optimal digital business outcomes.

Overview

Key Findings

- Rising business demands have increased reliance on blended teams consisting of in-house personnel and external contractors. Software engineering leaders often have weak strategies to manage them, resulting in suboptimal performance from both external resources and in-house software engineers.
- Software engineering leaders tend to use traditional project and productivity metrics, causing the wrong metrics to be tracked. They also struggle to give blended-team members visibility into their progress toward delivering intended outcomes.
- Generating business value requires a clear focus from multiple roles and blended teams across the delivery ecosystem. Many internal and external roles don't understand how their work connects with others' and contributes to business value.
- Blended teams often depend on other teams and the broader enterprise for resource allocation, spending, technology expertise and so on. Outcome attainment suffers without careful management of these dependencies.
- Engagements with software engineering vendors often drift into complacency, with external personnel becoming tactical and short-sighted instead of proactive and strategic.

Recommendations

Software engineering leaders tasked with managing blended teams should:

- Use the factors in Table 1 to choose an appropriate staffing model. Then, clarify roles and responsibilities with a RACI matrix, ensuring uniform visibility of this allocation across the blended team. Lastly, encourage the blended team to establish new software engineering standards and communication guidelines.
- Define a common set of metrics to measure the blended team's performance, including its overall effectiveness, product quality and contribution to business value.
- Create a climate of transparency by mapping the blended team's individual efforts to customer value.
- Reduce friction by managing blended-team interdependencies, including strategic, technical and governance-related interdependencies.
- Establish a culture of continuous improvement by creating a climate that fosters continual learning and provides continuous feedback.

Strategic Planning Assumption

By 2026, 80% of organizations that effectively combine insourced and outsourced software engineering teams through clear roles and responsibilities will see 30% improvement in team productivity.

Introduction

Apply Gartner's Five Principles for Managing Blended Software Engineering Teams

Software engineering leaders often struggle to achieve their digital business goals with a complex delivery ecosystem consisting of multiple blended teams. These teams can consist of a combination of in-house, cross-team and outsourced personnel (from one or more of the delivery options listed in Table 1).

Table 1: Delivery Options for Sourcing External Software Engineering Personnel

(Enlarged table in Appendix)

Type*	Definition	When to Use	Point of Control/ Responsibility	Level of Delivery Risk for Client
Staff augmentation (labor hire)	Expanding capacity temporarily to fill talent gaps by employing supplemental staff/temporary workers in moderate to high numbers	■ *Work priority:* Low ■ *Task requirements:* Unclear ■ *Length of engagement:* Short term	Client	High
Time and materials (T&M) consultants	Engaging consultants from management consultancies, outsourcers and systems integrators in moderate numbers on a T&M basis	■ *Work priority:* Moderate ■ *Task requirements:* Unclear ■ *Length of engagement:* Variable, though fixed in a few geographies (e.g., 18 months in the U.S.)	Client or service provider (sometimes)	Moderate
Direct-hire contracting (independent contractors and subcontractors)	Conducting a one-off recruitment, often in small numbers, to fill a specific, medium-term need	■ *Work priority:* Low ■ *Task requirements:* Clear ■ *Length of engagement:* Medium term	Client	Moderate
Gig workers, online platform freelancers or crowdsourcing workers	Using individual freelancers — or a group of them through an online platform — to solve specific problems or to perform specific tasks	■ *Work priority:* High ■ *Task requirements:* Clear ■ *Length of engagement:* Short term	Client or service provider (depending on engagement model)	Low to moderate

* This table lists four ways in which organizations can source software engineering personnel. Each of these sourcing options has a different intent, desired outcome and delivery method. No one method is better than the others. Each option carries its own set of risks as well. But largely, with any externally sourced option, there are general risks, such as loss of subject matter expert (SME) knowledge, necessity for high overtime compensation, inconsistent quality of resources, and complexity in contracts. Care should be exercised to provide contractual immunity against any such risks; this responsibility is more under the purview of the sourcing, procurement and vendor management leaders in the organization.

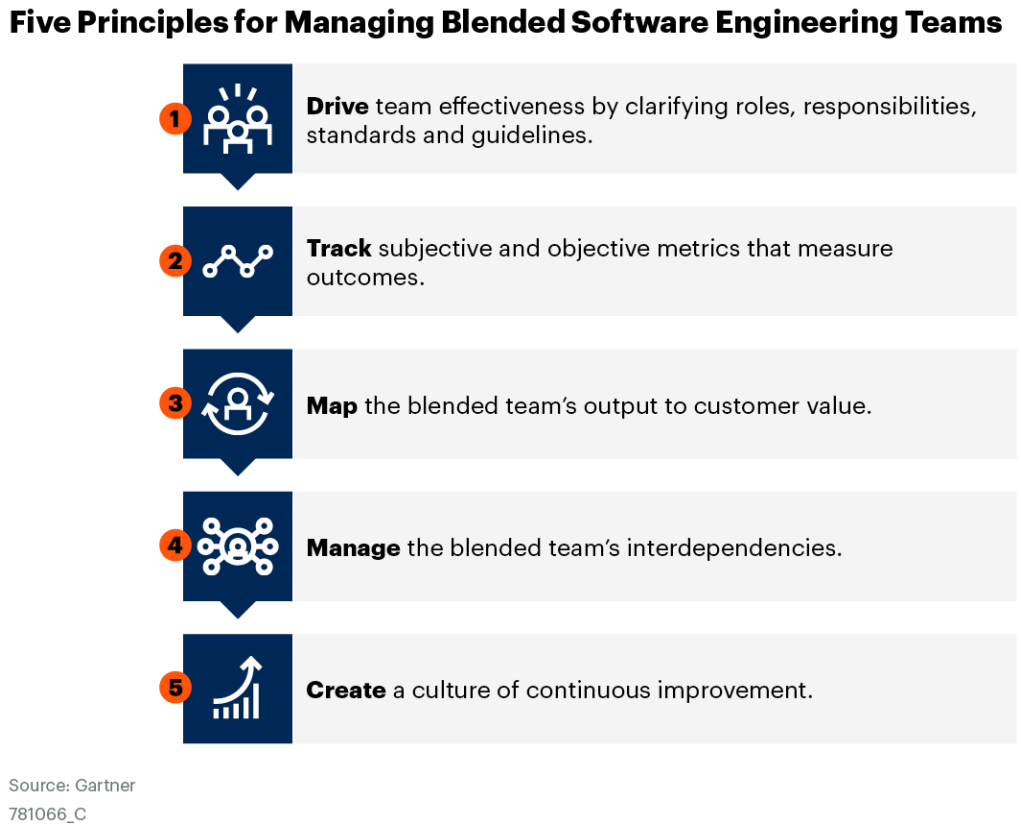
Source: Gartner (February 2023)

Gartner data shows that high-performing software engineering teams deliver 53% better outcomes in employee experience and productivity compared with low-performing teams, and 37% better outcomes in attracting and retaining external customers. (See Proven Strategies to Drive Software Engineering Team Effectiveness.) Therefore, it is imperative that software engineering leaders infuse this effectiveness into their teams.

To drive effectiveness in a multidisciplinary blended-team setup, where resources could come from different delivery options, software engineering leaders must seek new approaches, in addition to foundational improvement strategies like Agile, DevOps and automation.

To enable software engineering leaders to better manage these blended teams, Gartner recommends the five principles highlighted in Figure 1.

Figure 1: Five Principles for Managing Blended Software Engineering Teams



Source: Gartner
781066_C

Analysis

Principle 1: Select the Appropriate Staffing Option, and Clarify Roles, Responsibilities, Standards and Communication Guidelines

Agile, DevOps and automation are foundational improvement strategies because of their widespread adoption. They are critical to ensuring team effectiveness, and most software engineering leaders have room to further expand and mature adoption across their teams to gain the largest impact. However, these strategies alone are not enough for teams to achieve the highest levels of effectiveness, and software engineering leaders must seek new opportunities to help their teams continue to improve.

Actions for Software Engineering Leaders

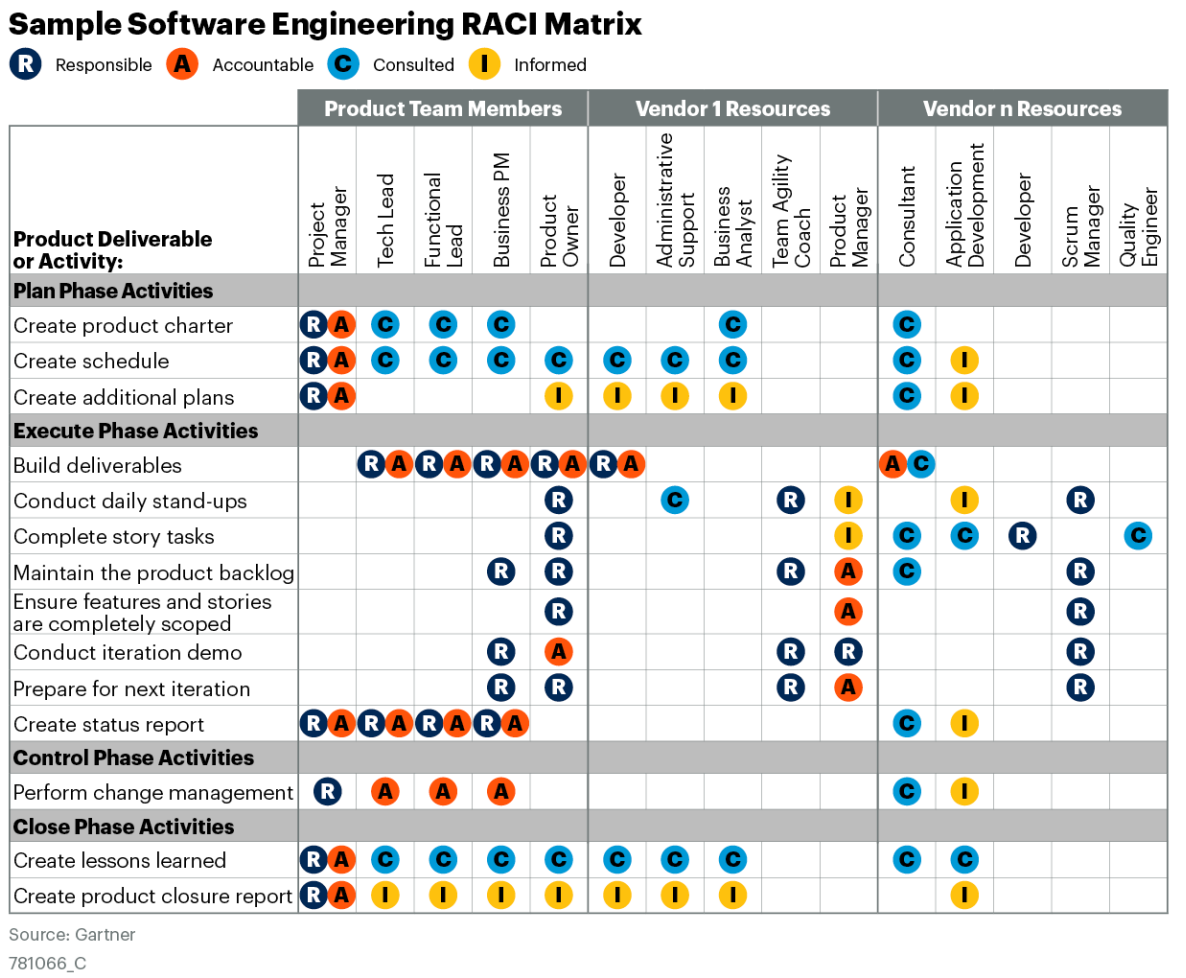
- Select the appropriate staffing model(s) from Table 1, based on work priority, task requirements, length of engagement, point of control and level of risk. This decision will require working collaboratively with sourcing teams to identify the options that best fit the organization's requirements and specific product scenarios.

- Define the set of business outcomes for which the blended team would be responsible/accountable. Use a responsible, accountable, consulted and informed (RACI) matrix (see Figure 2 for a sample RACI matrix for a typical blended team with resources from multiple vendors) . Use this matrix to not only clarify in-house roles and responsibilities, but also define what is expected from the vendor's personnel (shown as Vendor 1/n Resources in Figure 2) .

- Encourage the blended teams to shape new software engineering standards (see Note 1) specific to the outcomes they are trying to accomplish. This exercise helps increase team autonomy and also removes the stress of following overly restrictive or cumbersome standards that were historically set without the blended team's involvement (see Proven Strategies to Drive Software Engineering Team Effectiveness).

- Bring consistency to processes, tools and technologies. Use common definitions for defects, issues, acceptable criteria, and scope changes in coding and testing, thereby establishing a common line of communication for teams to adopt metrics-driven processes.

Figure 2: Sample Software Engineering RACI Matrix



Sample Software Engineering RACI Matrix

R Responsible A Accountable C Consulted I Informed

Product Deliverable or Activity:	Product Team Members					Vendor 1 Resources					Vendor n Resources				
	Project Manager	Tech Lead	Functional Lead	Business PM	Product Owner	Developer	Administrative Support	Business Analyst	Team Agility Coach	Product Manager	Consultant	Application Development	Developer	Scrum Manager	Quality Engineer
Plan Phase Activities															
Create product charter	R A	C	C	C				C			C				
Create schedule	R A	C	C	C	C	C	C	C			C	I			
Create additional plans	R A				I	I	I	I			C	I			
Execute Phase Activities															
Build deliverables		R A	R A	R A	R A	R A					A C				
Conduct daily stand-ups					R		C		R	I		I		R	
Complete story tasks					R					I	C	C	R		C
Maintain the product backlog				R	R				R	A	C			R	
Ensure features and stories are completely scoped					R					A				R	
Conduct iteration demo				R	A				R	R				R	
Prepare for next iteration				R	R				R	A				R	
Create status report	R A	R A	R A	R A							C	I			
Control Phase Activities															
Perform change management	R	A	A	A							C	I			
Close Phase Activities															
Create lessons learned	R A	C	C	C	C	C	C	C			C	C			
Create product closure report	R A	I	I	I	I	I	I	I				I			

Source: Gartner
781066_C

Gartner

Principle 2: Track the Metrics that Matter

Measuring the performance of software engineering teams has long been seen as a complicated and daunting task. This perception is becoming particularly acute as software becomes more complex and more decentralized. Software engineering leaders tend to adopt a project-by-project or a product-by-product focus while tracking metrics. This approach not only causes confusion, but also leads to a situation where the wrong metrics end up getting tracked or measured. Just because certain metrics *can* be measured doesn't mean they *should* be measured.

In Table 2, Column 2 provides a sample list of engineering and activity metrics that *do not* support developer well-being, engineering productivity and/or business performance. The issue of inappropriate metrics gets further compounded when you are managing a multidisciplinary fusion team consisting of in-house, cross-team and outsourced personnel. The lack of clarity around metrics not only leads to inconsistent expectations, but also obscures visibility into the progress being made and the amount of effort remaining. Some software engineering leaders also tend to divide up the metrics and separate them for in-house and outsourced development teams, making metric tracking a governance challenge eventually.

Actions for Software Engineering Leaders

- Define a common set of objective, yet meaningful, metrics to measure the performance of the blended team, including the team's overall health and effectiveness, product quality, and business value. Is something of value being delivered to partners, customers and the workforce? Table 2 provides a sample list of metrics that can answer this question.
- Use metrics to gain confidence (not certainty) that you will hit targets. Moreover, use them to drive improvement and value delivery internally, not to reward, punish or compare. Provide common visibility into the metrics to both in-house and external personnel. Map the metrics to the overarching yet primary measurement of success: business and customer value.
- Incorporate the defined metrics as part of the contract with the vendor. Ensure that the contract includes a mechanism to initiate remediation, such as a resource replacement process or a performance improvement plan, in the event of a major lapse in performance among the vendor-sourced personnel.

Table 2: Sample List of Engineering Activities to Measure Metrics

(Enlarged table in Appendix)

Number of Activity	What Not to Measure	What to Measure	What Does It Measure?
1	Hours worked	Team health	Happiness, morale, collaboration and improvement appetite
2	Lines of code	Code coverage	Testing effectiveness
3	Bugs fixed	Escaped defects	Number of defects identified after release date
4	Tasks completed	Cycle time	Elapsed time of a task, from start to finish
5	Number of commits	Deployment frequency	How often code is deployed into production
6	Velocity points	Velocity	Average amount of work completed during a sprint
7	Pull request count	Change failure rate	Number of failures per number of deployments to production
8	Features shipped	Lead time	Total time taken for work to move through the value stream, from request to delivery
9	Impact scores	Customer satisfaction score	How satisfied or happy the customer is with the final delivery
10	Story points per hour	Release burndown	Progress of a release, with an indication of remaining effort

Source: Gartner (February 2023)

Principle 3: Create a Climate of Transparency by Mapping Individual Contributors' Efforts to Customer Value

Software engineering leaders often struggle to align the efforts of multiple roles and teams across the delivery ecosystem to maximize value. Many roles often do not have direct visibility into how their work connects with others' and contributes to the value ultimately delivered to the business. Delivery activities, processes and goals are siloed, making it difficult for different roles and teams to understand how their work connects and ultimately contributes to customer impact and value.

These disconnects get compounded further when each team includes representatives from external vendors. Lack of visibility into how value is created can cause delays or erode value delivery. For example, these disconnected teams may not have clarity on how to contribute, or they may fail to prioritize the most critical needs.

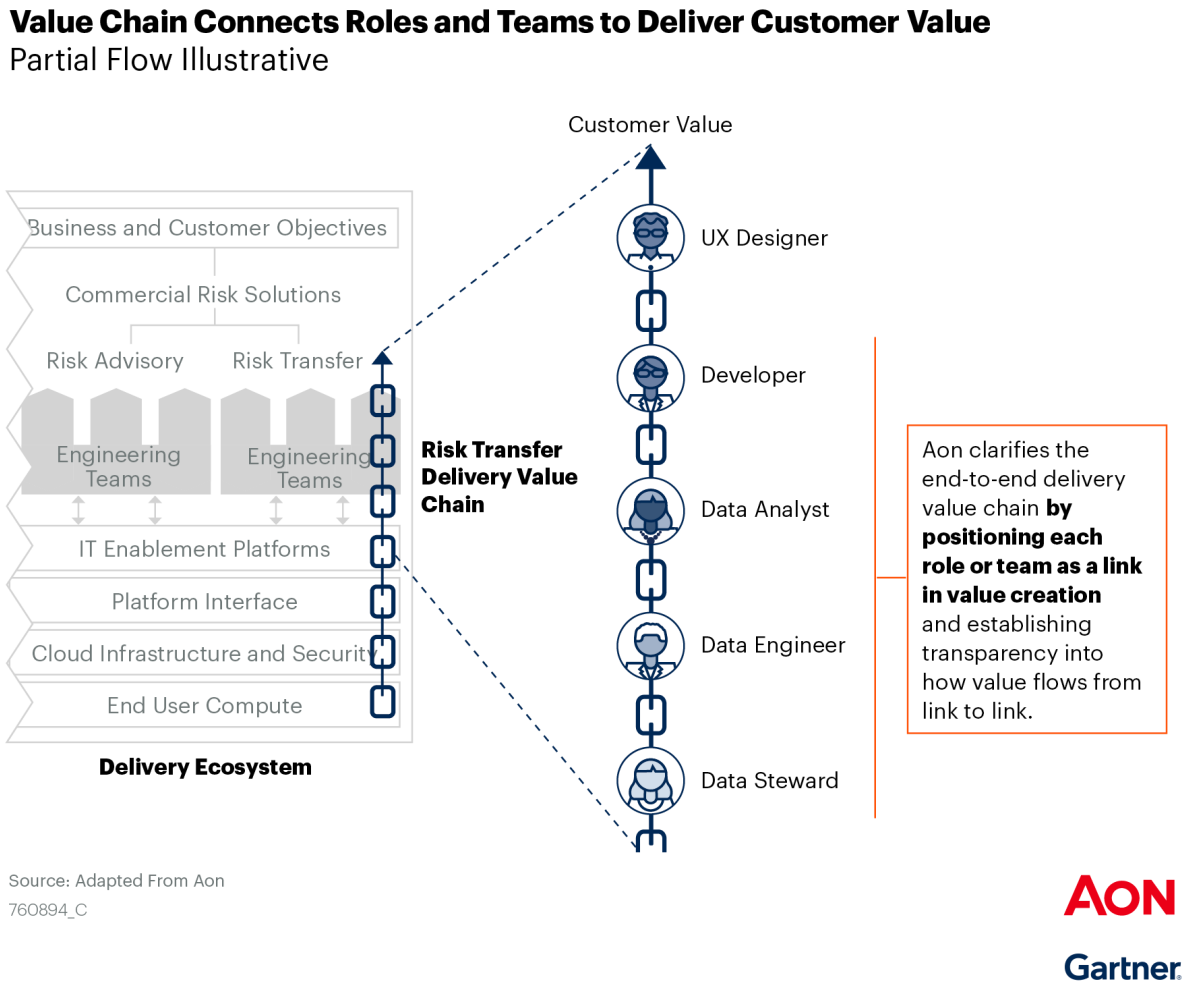
Case in Point: Establishing Value Chains (Aon)

Aon approached and remediated the above conundrum by establishing value chains (see Figure 3). By establishing value chains to clarify the roles and teams involved in its end-to-end series of activities, Aon shortened time to value by 50% over the last three years.

The shift to delivery value chains has also helped Aon improve customer satisfaction by fostering a customer-centric mindset in all roles and teams involved in solution delivery. This orientation enhances the value each role produces for the next link in the value chain and ultimately for the end customer.

For the full case study, see Case Study: Align Product Delivery to Customer Value Using Value Chains (Aon) .

Figure 3: Value Chain Connects Roles and Teams to Deliver Customer Value



Actions for Software Engineering Leaders

- Establish similar value chains to those in the Aon case study, tailoring them to your organization's structure. Work closely with business partners and enterprise architects to first define the customer value you want to achieve. Use this customer value as the benchmark for the different teams and roles in the value chain.
- Break down the activities required to achieve customer value, and map them to the chain of roles and teams that perform them across the delivery ecosystem. This exercise defines the value chain and provides transparency into how different teams and roles come together to deliver customer value.
- Clarify each role's contribution to value by orienting each role or team as a link that produces and consumes value from other links in the value chain. Help each link understand how it builds on the value from the preceding link to create value for the subsequent link and, ultimately, for the end customer.

- Regularly monitor the time taken to deliver value by each team and by the overall value chain to identify and address inefficiencies.

Principle 4: Reduce Friction Caused by Internal and External Team Interdependencies

Despite the hype about autonomous, Agile teams, the work of blended multidisciplinary teams is highly connected with the work of other teams within and beyond blended teams. Software engineering leaders must help blended teams manage interdependencies — in relation to strategy and architectural alignment, for example — to improve these teams' alignment with enterprise goals and to accelerate desired outcomes. These interdependencies are defined as follows:

- **Strategic:** Blended teams depend on enterprise strategies, business unit strategies and the work of other fusion teams to direct their work and prioritize their resources.
- **Governance:** Blended teams depend on inputs or approvals from subject matter experts for various activities (secure coding and release management, for example).
- **Skills:** Blended teams may sometimes lack key skills and capabilities (such as expertise in user experience or quality assurance) needed to meet their objectives. These skills gaps increase the blended teams' reliance on other teams or resources.
- **Technology:** Many technologies span multiple domains and cannot be aligned to a single fusion team. Thus, fusion teams need to coordinate delivery schedules with other teams that may have different priorities.

Actions for Software Engineering Leaders

- Work with the CIO and other CxOs to refocus structures and roles from traditional hierarchies to product management approaches that intentionally support alignment with enterprise strategies. Redesign role families, add new roles where necessary, and promote cross-team movement to ensure that IT and business employees can better serve the needs of a product management organization.
- Partner with key IT and business stakeholders to define objectives and key results (OKRs) for enterprise programs that cut across blended-team silos. Ensure different teams adopt OKRs to align with enterprise strategy.
- Embed enterprise experts, such as enterprise architecture (EA), legal and compliance staff, in blended teams to provide "in the moment" support, thus reducing avoidable handoffs and escalations beyond these teams.

- Minimize technical interdependencies by providing blended teams with reusable technology components that embed architecture and security controls by design. Build support for a composable platform strategy by articulating its impact on business priorities. For example, benefits include a consistent employee experience or customer experience, better security, and greater efficiency through reuse.

Principle 5: Create a Culture of Continuous Improvement

Software engineering leaders cannot simply take a clever combination of talented in-house, cross-team and outsourced personnel, form a blended team with them, and then hope that it delivers. They must drive high performance within their teams by creating a culture of continuous improvement. A culture of continuous improvement is essential to keep blended teams effective. It requires continuous learning on what works well and what does not.

Actions for Software Engineering Leaders

- Regardless of whether the product owners are in-house or vendor-supplied resources, conduct regular retrospectives (see Note 2) with their respective blended teams on key iterations. At the same time, encourage the product owners to involve stakeholders from the wider organization, such as sales, marketing and customer success, in postmortems (see Note 3) .

- Encourage blended team members to gather new ideas on software/product development (or on whichever area their expertise lies) from outside the organization by networking with external peers. Create learning programs that can help blended team members get up to speed on best practices and latest developments.

- Provide process visibility, and highlight areas that need coaching by implementing common metrics dashboards with business analytics. Use flow metrics to identify opportunities to improve delivery performance (see How Software Engineering Leaders Can Use Value Stream Metrics to Improve Agile Effectiveness).

- Celebrate bold conversations, and work toward eliminating ambiguity and mismatched expectations. Express appreciation for in-house employees' and contractors' willingness to vocalize questions, doubts and confusion, and help them determine the next best steps. (See Quick Answer: How Do We Build Psychological Safety in Our Software Engineering Teams.)

- Work with your vendors to define a program of continuous improvement. Identify improvement milestones on a quarter-over-quarter basis (or whatever frequency suits your organization's pace). Incentivize your vendors by giving them a glimpse into the future work programs that could come their way if their personnel do a good job in accomplishing all the improvement milestones.

Evidence

2022 Gartner Global Labor Market Survey: Some of the data cited in this research is from Gartner's 3Q22 Global Labor Market Survey. The survey was based on responses from 18,010 employees globally, including 1,717 employees in the IT function. Responses were collected monthly across 40 different countries in 15 languages, and were then aggregated to generate quarterly findings. There are no statistically significant differences in the sample composition across the three months.

Note 1: Definition of a Software Engineering Standard

A software engineering standard is a standard, protocol or other common format of a document, file or data transfer. It is accepted and used by one or more software developers while working on one or more than one computer programs. It consists of certain terms, concepts, data formats, document styles and techniques that are agreed upon by software creators so that their software can understand the files and data created by a different computer program. Software standards enable interoperability between different programs created by different developers.

Note 2: Definition of a Retrospective

A retrospective is a meeting that's held at the end of an iteration (sprint) in Agile software development. During the retrospective, the team reflects on what happened in the iteration and identifies actions for improvement going forward.

Note 3: Definition of a Postmortem

A postmortem is a full-scope review of a project, delivered after the project's completion. It isn't usually performed as part of Agile frameworks.

Disclaimer: The organization (or organizations) profiled in this research is (or are) provided for illustrative purposes only, and does (or do) not constitute an exhaustive list of examples in this field nor an endorsement by Gartner of the organization or its offerings.

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

How to Manage Blended Teams of In-House and Outsourced Software Engineering Personnel

Case Study: The Estée Lauder Companies Is Transforming Its IT Services Sourcing for Digital Delivery

Leader Guide to Fostering Psychological Safety During a Crisis

Use the Right Metrics in the Right Way for Enterprise Agile Delivery

Software Engineering Teams Must Learn to Deliver More Value

How to Lead High-Performing Software Engineering Teams

Contracting for IT Contingent Labor Services

How to Manage Vendor Performance Risks in Agile Projects

Table 1: Delivery Options for Sourcing External Software Engineering Personnel

Type*	Definition	When to Use	Point of Control/ Responsibility	Level of Delivery Risk for Client
Staff augmentation (labor hire)	Expanding capacity temporarily to fill talent gaps by employing supplemental staff/temporary workers in moderate to high numbers	■ *Work priority*: Low ■ *Task requirements:* Unclear ■ *Length of engagement:* Short term	Client	High
Time and materials (T&M) consultants	Engaging consultants from management consultancies, outsourcers and systems integrators in moderate numbers on a T&M basis	■ *Work priority:* Moderate ■ *Task requirements:* Unclear ■ *Length of engagement:* Variable, though fixed in a few geographies (e.g., 18 months in the U.S.)	Client or service provider (sometimes)	Moderate
Direct-hire contracting (independent contractors and subcontractors)	Conducting a one-off recruitment, often in small numbers, to fill a specific, medium-term need	■ *Work priority:* Low ■ *Task requirements: C*lear ■ *Length of engagement: M*edium term	Client	Moderate

Gig workers, online platform freelancers or crowdsourcing workers	Using individual freelancers — or a group of them through an online platform — to solve specific problems or to perform specific tasks	■ *Work priority:* High ■ *Task requirements:* Clear ■ *Length of engagement:* Short term	Client or service provider (depending on engagement model)	Low to moderate

* This table lists four ways in which organizations can source software engineering personnel. Each of these sourcing options has a different intent, desired outcome and delivery method. No one method is better than the others. Each option carries its own set of risks as well. But largely, with any externally sourced option, there are general risks, such as loss of subject matter expert (SME) knowledge, necessity for high overtime compensation, inconsistent quality of resources, and complexity in contracts. Care should be exercised to provide contractual immunity against any such risks; this responsibility is more under the purview of the sourcing, procurement and vendor management leaders in the organization.

Source: Gartner (February 2023)

Table 2: Sample List of Engineering Activities to Measure Metrics

Number of Activity	What Not to Measure	What to Measure	What Does It Measure?
1	Hours worked	Team health	Happiness, morale, collaboration and improvement appetite
2	Lines of code	Code coverage	Testing effectiveness
3	Bugs fixed	Escaped defects	Number of defects identified after release date
4	Tasks completed	Cycle time	Elapsed time of a task, from start to finish
5	Number of commits	Deployment frequency	How often code is deployed into production
6	Velocity points	Velocity	Average amount of work completed during a sprint
7	Pull request count	Change failure rate	Number of failures per number of deployments to production
8	Features shipped	Lead time	Total time taken for work to move through the value stream, from request to delivery
9	Impact scores	Customer satisfaction score	How satisfied or happy the customer is with the final delivery

10	Story points per hour	Release burndown	Progress of a release, with an indication of remaining effort

Source: Gartner (February 2023)

Actionable, objective insight

Position your organization for success. Explore these additional complimentary resources and tools for software engineering leaders:

eBook

Build a World-Class Software Engineering Organization

Gain insights on the three key drivers of success over time for software engineering.

Download Now

Webinar

Improve Developer Productivity, Retention and Morale With Platform Engineering

Optimize the developer experience and accelerate the delivery of customer value.

Watch Now

Research

How to Attract, Develop and Retain Great Software Engineering Talent

Build a compelling employee value proposition and agile learning environment.

Read Now

Peer Community

Software Engineering Leaders

Ask questions and add polls to get fast answers from your peers.

Join Now

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

Connect With Us

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

U.S.: 1 855 811 7593

International: +44 (0) 3330 607 044

Become a Client

Learn more about Gartner for Software Engineering Leaders
gartner.com/en/information-technology

Stay connected to the latest insights

Attend a Gartner conference
View Conference

Gartner®