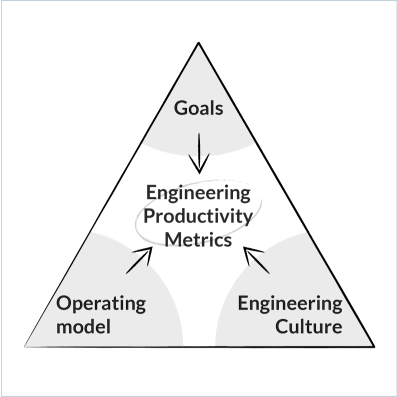
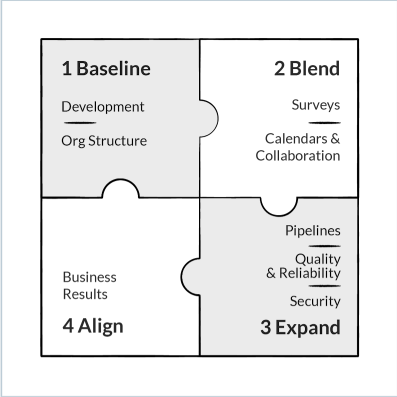
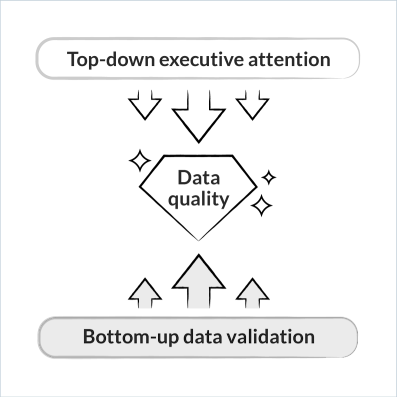
FAROS

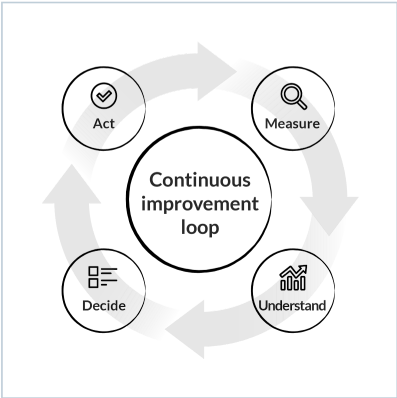
The Engineering Productivity Handbook

A guide to tailoring your initiative to your goals, operating model, and culture.









Introduction

Why Engineering Productivity Matters

Engineering is an increasingly important and expensive function. According to BCG, the fastest-growing companies are spending more than 20% of their revenues on R&D and as much as 40% to 50% when trying to expand beyond their core products.[1]

Engineering leaders are being asked to know their business thoroughly and explain what engineering is doing, how it relates to key company initiatives, and what resources they need and where. To say it simply, they must be able to conduct business-oriented conversations about very technical things.

Those answers are usually knowable for a small shop with a score of engineers, but as you scale, it gets much harder. That's when organizations typically establish engineering productivity programs to create a line of sight for leaders while, in parallel, empowering line managers and teams to make the right decisions.

The Faros AI Approach

Software engineering intelligence platforms like Faros AI provide visibility into engineering productivity, a catchall phrase for efficiency and effectiveness.

Some vendors will tell you there's a right way and a wrong way to measure productivity. The Faros AI approach is more nuanced. We say: **There is a right way, and it depends on your context.**

> Your engineering productivity program should be adapted to your context, including what you need to achieve, how you work, and what you value.
>
> Your context will naturally change over time as you grow, evolve, and respond to market forces, and your program will evolve along with these changes.

The modularity, customizability, and extensibility of the Faros AI platform were intentionally designed to support this reality. You'll learn why these features are important as you read on.

About this Guide

This guide will help you identify the right path for you today and navigate the five main steps of a successful data-driven engineering productivity program.

[1] How Software Companies Can Get More Bang for Their R&D Buck, November 22, 2019, bcg.com

This is how we've built this guide:

01 What to measure

02 How to collect the data

03 How to normalize and validate the data

04 How to analyze the data

05 How to operationalize the data to achieve impact

We recommend you revisit the guide periodically to reorient your program as your context changes.

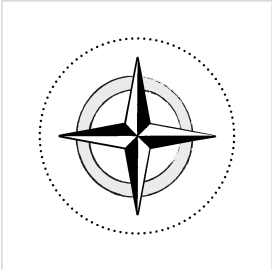
Chapter 1: **What to Measure**

The SPACE framework is the most comprehensive framework for measuring engineering productivity today. We like SPACE because it advocates for a holistic view of productivity without being overly prescriptive. It also combines system-generated telemetry with developer sentiment gathered from surveys and interviews. Measuring multiple dimensions shields your program from unintended consequences and potential metric gaming. (If you're familiar with DORA metrics, they are a subset of the SPACE framework.)

But adopting SPACE in reality is hard. SPACE offers suggestions for a smorgasbord of metrics in each of its five dimensions: **S**atisfaction and well-being, **P**erformance, **A**ctivity, **C**ommunication and collaboration, and **E**fficiency and flow. Which metrics should you pick? Which ones are right for you?

In this chapter, we'll answer these questions:

- Which metrics are important to measure?
- What data points help us understand the health of our engineering organization?



Guiding Principle: Identify What Matters to You

Before deciding what to measure, take a beat to identify what's important to you, how you define success, and what productivity looks like to you.

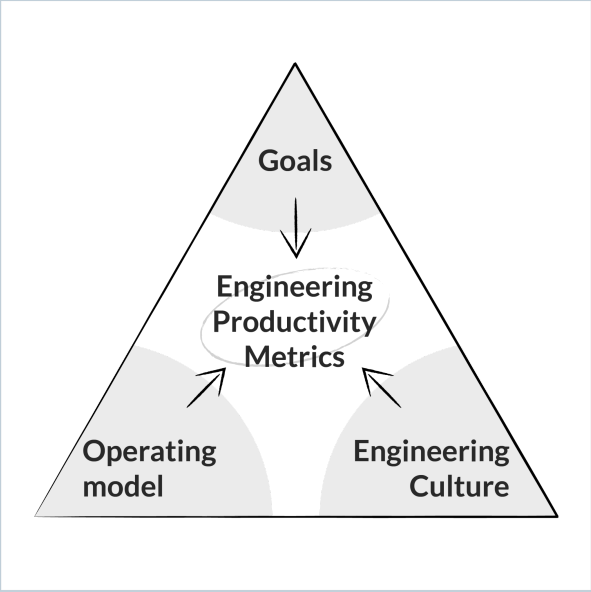
We've prepared a 3-step guide to help you answer these questions. As you read, mark the rows that sound like you. At the end of this exercise, you'll have a good idea of the leading and lagging indicators to measure, the dimensions for analysis, and the way the information will be shared.

In selecting what to measure, consider three elements:

- **What you need to achieve:** Companies have different goals, priorities, and challenges at different stages of their lifecycle. A scaling ecommerce startup that just raised its Series C is optimizing for different results than a publicly traded enterprise in a highly regulated domain. This guide will help you identify what to measure at your current stage.[2]
- **How you work:** Your operating model introduces different dimensions to how you analyze and examine productivity. Do you insource or outsource? Are you hybrid or fully remote? Are you structured with regional hubs or geo-concentrated? Do you have multiple SDLCs?
- **Your engineering culture:** The organization's values and DNA determine which data is collected. For example, a company that uses ranking in its formal performance management processes will likely be comfortable with individual metrics, while others may prefer to go down to the team level only.

We've prepared a step-by-step guide to help you answer these questions. As you read, mark the rows that sound like you. At the end of this exercise, you'll have a good idea of the leading and lagging indicators to measure, the dimensions for analysis, and the way the information will be shared.

Figure 1.1: Business Context that Influences Metric Selection



[2] Large organizations will likely have different internal groups facing very different challenges. Your corporate strategy can help inform which areas to focus on first.

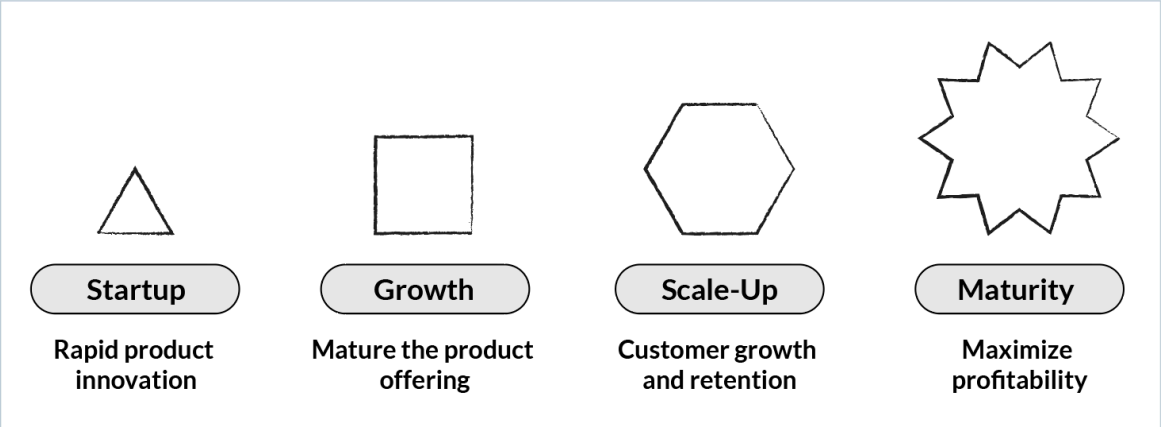
Step 1: Identify Your Goals

Below are typical goals and engineering productivity metrics based on the stage of your company. Metrics are additive as a company progresses through the stages.

Table 1.1: Common metrics aligned to goals per company stage

Stage	Goal	Common Metrics
Startup	**Rapid Product Innovation** Remove blockers to launching new features and finding product-market fit.	Lead time and cycle times Throughput Deployment frequency % delivered vs. committed Bottlenecks
Growth/ Expansion	**Mature the Product Offering** Develop a technical strategy to support an expanding tech stack and team. Introduce the scaffolding to more teams, a larger codebase, and dependencies.	***All of the above, plus:*** Production stability (uptime, MTTR) Platform/infra effectiveness Cross-team dependencies Code quality (coverage, test stability, smells, security) Team productivity comparisons and benchmarks Team composition Developer satisfaction Revenue per R&D FTE
Scale-up	**Customer Growth and Retention** Balance speed with quality, safety, and reliability to support a growing customer base.	***All of the above, plus:*** On-time roadmap delivery Velocity and quality benchmarks (DORA 4) SLO and SLA compliance Security and privacy compliance Resource allocation vs. ideal targets Onboarding effectiveness Developer wait time (e.g., Git performance, build time, CI reliability, test flakiness) R&D OpEx per R&D FTE
Maturity	**Maximize Profitability** Reduce costs, standardize to industry standards, and improve retention.	***All of the above, plus:*** Initiative tracking and impact Infrastructure costs Migration and consolidation impact metrics Individual and team performance Talent and skill composition

Figure 1.2: The Evolving Engineering Goals for Companies



Step 2: Identify Your Operating Model

Below are typical operating models that indicate the additional lenses through which you'll want to analyze your metrics.

Table 1.2: Common metrics and analysis dimensions per operating model

Operating Model	Description	Common Metrics
Heavily outsourced	Your organization relies on sub-contractors, usually from more than one vendor. Your metrics should help **compare insourced to outsourced** productivity, measure the value you receive from each vendor, and ensure institutional knowledge is being captured to prevent vendor lock-in.	**Productivity metrics per contract type and vendor** Productivity per dollar spent Activity per dollar spent Time spent vs. target hours Velocity and throughput Lead time and cycle times Active vs. waiting times with special attention to handoffs and approvals Quality of delivery (e.g., bugs per task) Code, test, and documentation coverage Task and PR hygiene
Geographically distributed	Your organization has globally distributed development centers. Your metrics should help assess the **relative productivity of each location** and identify collaboration challenges that must be addressed.	**Productivity metrics per location** Productivity per dollar spent per location Impact of cross-geo collaboration on velocity, throughput, and quality metrics Impact of cross-geo collaboration on MTTR and SLAs

Operating Model	Description	Common Metrics
Remote/Hybrid	Your organization has multiple employment types, including in-person, hybrid, and remote developers. Your metrics should help **assess the relative productivity of each employment type**. They should measure the impact of work-from-home policies on productivity and engagement and help inform policy, hiring, and promotion decisions.	**Productivity metrics per employment type** Onboarding effectiveness per employment type The 'before and after' impact of WFH policy changes Developer experience and satisfaction per employment type
Centralized SDLC	Often characterized by a monorepo, a centralized SDLC has specific impacts on the developer experience that need to be tracked. Your metrics should help you **identify technical areas for optimization**. They need to be sliced and diced by application or services (as opposed to repo) and pinpoint where dependencies are slowing developers down.	**Productivity metrics per application or service** PR review SLOs Commit queue SLOs Remote build execution and cache SLOs Clean vs. cached build volume and runtimes Test selection efficacy based on compute resources and change failure rate
Multiple SDLCs	Your organization has multiple SDLCs, often a result of a large portfolio and M&A. Your metrics should help **identify high-performing SDLCs** to increase the cross-pollination of best practices and reduce the duplication of efforts.	**Productivity and experience comparison per SDLCs**

Step 3: Identify Your Engineering Culture

Corporate and engineering culture will also influence the smallest unit of measurement, whether the individual or the team and how the metrics are applied. The table below illustrates a few examples.

Table 1.3: Common metrics and analysis dimensions per engineering culture

Culture Type	Description	Metrics
Compete culture	Employees are evaluated based on strict performance metrics	Productivity by level and against forced distribution

Culture Type	Description	Metrics
	and stack-ranked periodically.	Productivity vs. ideal (expectations)
Family culture	Engineering is encouraged to push towards a collective goal.	Productivity at the team level (not the individual)
Ad hoc culture	New teams are frequently spun up to collaborate on shorter-term projects.	Productivity by collaboration unit (e.g. project, application, squad) as defined by GitHub teams or Jira boards
Decentralized culture	Self-managed autonomous teams are provided a budget and are accountable for delivering business results.	P&L metrics

Once you've identified the common metrics associated with your goals, operating model, and culture, you'll then need to collect the relevant data.

Chapter 2: **Collecting the Data**

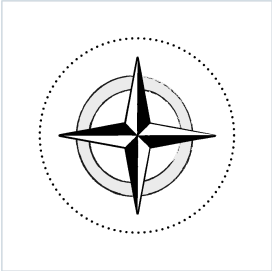
Collecting data is the first step to creating visibility. Organizations commonly face the challenge of "islands of information," where data is artificially siloed according to tool specialization. Cross-tool and cross-domain analysis often requires contorting and manipulating exported data into spreadsheets.

With Faros AI, you can retire your spreadsheets. Your metrics will be generated from a rich and complex combination of data sources, some standard and some bespoke, covering SaaS products and homegrown solutions, org structure data from HR systems, developer experience surveys, cost data from business systems, and much, much more.

In this chapter, you'll learn a field-proven strategy for collecting data incrementally, creating valuable insight into productivity at each stage.

In this chapter, we'll answer these questions:

- What sources should we extract engineering data from, and where do we start?
- Which data sources help tie engineering work to business value?
- Should the data we collect be quantitative, qualitative, or both?



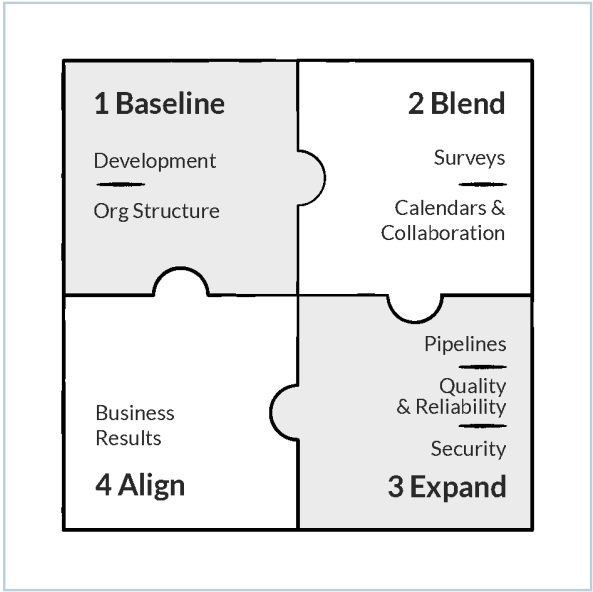
Guiding Principle: Begin with the Basics and Advance

In Chapter 1, you identified the metrics that can support your productivity goals and the secondary data sources that will allow you to analyze them based on your operating model. You also thought about how this data will be consumed and shared based on your engineering culture.

So now's the time to begin assembling those pieces. A step-wise approach can deliver quick wins, build trust, and gradually develop the data-driven mindset you need. Layer by layer, you will assemble the complete picture of engineering productivity as you've envisioned it.

The illustration below shows how to gradually create a complete picture of engineering productivity. So let's break it down.

Figure 2.1: The step-wise approach to data collection



Step 1: Baseline

Set aside concerns about data quality and data hygiene. Normalization and validation address those (we'll talk about this more in Chapter 3) and are not a barrier to collection. The first step involves baselining the current state in support of your first or primary use case.

- Most organizations start with basic questions about developer productivity: How are engineers spending their time, how long do cycles take, and where are the bottlenecks? To that end, **connect to data sources for tasks and PRs.**
- To support how you want to slice and dice the information based on your operating model, you'll also need to collect the relevant metadata about your reporting structure (and occasionally, your product or app hierarchies). **Connect HR data like Workday or import the data from an internal metadata service.**

Step 2: Blend

Developer surveys capture developers' perceptions of how their team delivers. They provide insights into points of friction in the software delivery process and more descriptive feedback about what can be improved at the team or organizational level. Developer surveys are key to tracking employee engagement and satisfaction with the developer experience over time.

With your intelligence platform in place, the detailed and highly contextual feedback from developers can be lined up against the data you've collected from engineering systems and processes. Powerful insights come from blending qualitative insights from surveys with telemetry about systems, processes, and workflows. You'll also discover the next set of high-priority data sources to connect to for deeper analysis.

- **Collect the results of developer surveys** onto the platform, such that sentiment data from employee responses can be intersected with telemetry-based data from engineering systems. This helps corroborate specific complaints about speed, complexity, and dependencies with supporting systems data.
- Developers frequently complain that they spend too much time in meetings. **Collect information about meeting load from collaboration systems like Calendars.**

Step 3: Expand

Don't lose sight of the KPIs that act as checks and balances within your initiative. According to the authors of the SPACE framework, "productivity cannot be reduced to a single dimension… Only by examining a constellation of metrics in tension can we understand and influence developer productivity."

That's why, at this stage, you'll want to expand to data sources related to quality and reliability to prevent over-focusing on velocity. You've likely identified several of these metrics in Chapter 1, and you're now ready to generate them now that the basics are in place.

- **Collect data to form a holistic view of quality, reliability, and security.** Pre-production sources provide a view into deployments, bugs, code smells, vulnerabilities, code and test

coverage, test stability, and flakiness. Post-production sources are the systems of record for customer issues and tickets, production incidents, and system outages.

- **Collect service-level metrics** to correlate “code production" metrics with the stability and efficiency metrics of supporting systems and platforms. This includes metrics like CI reliability, build cache hit ratio, development environment usage statistics, and more.

Step 4: Align

The C-Suite expects the engineering department, like every other corporate function, to demonstrate its impact on corporate objectives. To that end, the next step is collecting business results data in support of quarterly and annual planning and OKR tracking.

Some of that information may be readily available in the task management systems you’ve connected, which will allow you to measure say/do ratios and on-time delivery of product roadmap.

That said, there is a lot more opportunity to illuminate costs and impacts by intersecting engineering data with business metrics pertaining to product usage, customer satisfaction, and financial performance.

- **Collect data from systems that record targeted business results**, like cloud costs, uptime stats, customer satisfaction, and efficiency stats for business processes supported by engineering.
- For engineering teams that operate in lower layers of the stack, **consider proxy metrics that represent value to those who consume their output.**

The following table summarizes the data sources to connect at each of stage of your program, with examples of specific tools.

Table 2.1: Data Sources for Engineering Productivity

Stage	Sources	Examples
Start	Task management systems Source control systems CI/CD build and deploy events HR data	Jira, Asana, ADO GitHub, BitBucket Workday
Blend	Survey tools and spreadsheets Calendars	Google Sheets, Airtable, Qualtrics Google Calendar
Expand	Code quality tools CI/CD build and deploy events CI/CD individual test events Incident management tools	SonarQube Jenkins, CircleCI, Spinnaker, ArgoCD ServiceNow, PagerDuty, StatusPage

Stage	Sources	Examples
Align	Financial systems Customer experience Product analytics	Salesforce Gainsight Amplitude

In the next sections, we'll provide guidance on the practicalities of collecting data from different data sources, including:

- Standard and custom data sources
- Sources for reporting structure
- Sources for data attribution and ownership
- Sources that require schema extension

Standard and Custom Data Sources

During the data collection phase, the data will be stored in the Faros AI canonical schema, where it is normalized into a single, connected data set that can be queried efficiently across the entire organization.

Inevitably, your data sources will be a combination of vendor tools and homegrown sources, handled as follows:

- **Popular vendor tools**. Faros AI leverages optimized open-source connectors for common task management tools like Jira and Asana, source control tools like GitHub and GitLab, static code analysis tools like SonarQube and Codacy, and others. Connectors typically pull historical data from the source **once** and then pull dataset changes on a cadence. When possible, we recommend also using Webhooks to push data to Faros in real-time.
- **Custom homegrown tools and sources (including spreadsheets).** Two easy options exist to ingest data from custom sources.
    - **Connector Development Kit:** Faros AI leverages an open-source framework called Airbyte, which has connectors to common sources like S3 and Postgres to ease development. You'll simply have to map the data from the custom source to the schema.
    - **Events CLI:** For builds and deployments, it is easy to instrument the tool and have it send events to Faros AI. This is accomplished by adding a line of code to a preexisting CI/CD script.

Sources for Reporting Structure

Engineering productivity data is often examined based on reporting structure, with leaders tracking metrics for the sub-org and teams they manage. This essential information can be ingested from an HR source like Workday, which will update Faros AI when people join or leave the company or upon a major re-org.

Within Faros AI, the reporting structure is used for rollups and drill-downs because it represents how your teams, teams of teams, and groups are organized. It also enables comparisons, outlier identification, and team-tailored insights.

Sources for Data Attribution and Ownership

In addition to the formal reporting hierarchy, Faros AI infers the mapping between teams, apps, and repos. (While rare, if there is a source of truth for this data, it can be ingested directly instead.)

Furthermore, for any specific metric, your organization can choose whether to attribute it to a team based on:

- the people whose data contributed to the work *or*
- the specific app or repo that the data touches

Faros AI will auto-select the best attribution method based on experience and domain expertise, but as with any other metric behavior, this is configurable.

Extending the Schema for Special Use Cases

The Faros AI schema is designed to represent all the relevant SDLC data in a cohesive, interconnected manner. To use it, you simply need to plug in your data.

The standard schema will be sufficient in 99% of cases. In rare edge cases where a certain data set does not fit perfectly within the schema, the schema can be extended by leveraging tags, custom columns, and custom metrics. If your use case is of general interest, Faros AI will consider making it a first-class concept in the schema.

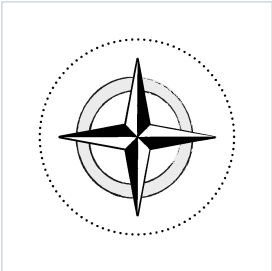
Chapter 3: **Normalizing and Validating the Data**

Every sizable engineering organization inevitably has different teams working in different ways. They have different workflows running on different tools and pipelines, use different custom fields and statuses, and release on different cadences.

To analyze all this data through a single lens requires normalization, and to trust the data requires confidence in its quality. We've worked with some of the world's most complex software engineering organizations to develop a playbook to achieve both.

In this chapter, we'll answer these questions:

- How should we deal with multiple SDLCs and variance in tool usage?
- How should we deal with poor data quality or data hygiene?



Guiding Principle: Don't Let Perfect Be the Enemy of Good

To put it bluntly, every organization has data hygiene issues when it comes to human-curated data. That's the bad news. But here's the good news:

- Your organization has a lot of clean machine-generated data about PRs, builds, and deployments, where data hygiene is not an issue.
- Faros AI does more than enough normalization to baseline performance and highlight hotspots, even with less-than-perfect hygiene for human-curated data.
- In the early days of your program, you'll be focused on coarse, high-level metrics that baseline and benchmark performance and highlight areas of concern. This is similar to the

output of an audit or a consultancy engagement. At this level, the law of large numbers makes hygiene less of an issue.

The guiding principle here is to start measuring to create high-level visibility and then let the teams identify and address pressing data hygiene issues once they see how they are skewing their metrics. Never ask teams to change how they manage projects or use Jira before they start actually generating metrics and insights. Instead, let Faros AI highlight the inconsistencies that teams should address.

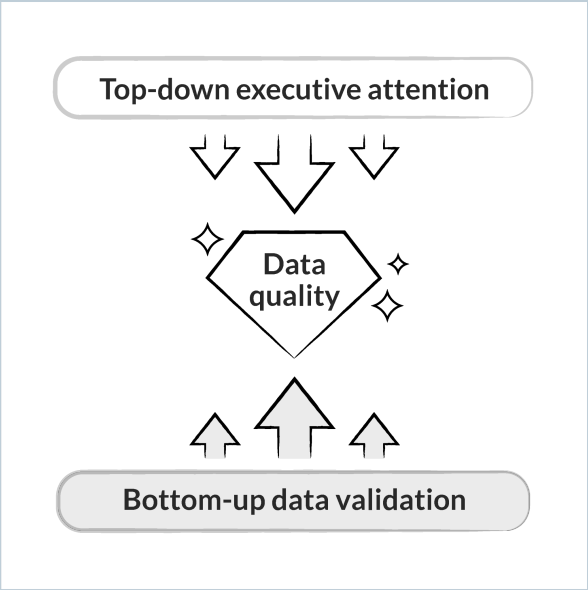
> *There will always be data hygiene problems. But if you can get a smaller set of clean data into Faros AI—for us it was PR data and build data—you can get those initial benchmarks. They you can move on to improving data hygiene because you've built a data mindset in your engineering teams.*
>
> — Tulika Garg, Director of Developer Enablement & Ecosystem, Autodesk

Good data quality results from the organization's commitment to becoming data-driven. Normally, no one is incentivized to address these issues until they start impacting highly visible metrics. That is to say, once leaders start paying attention to metrics, the data hygiene issues will be fixed—but not before.

The road to improving data quality involves top-down and bottom-up motions:

- **Top-down:** Leaders hold their organization accountable for important metrics.
- **Bottom-up:** Teams review their metrics and address any underlying data quality issues.

Figure 3.1: Data quality improvement through top-down and bottom-up motions



If you treat data quality as a showstopper, you will never get visibility. And without visibility, you'll never address the data quality issues. You'll waste years trying to address data hygiene only to discover you were focused on fixing the wrong things.

In summary, here are some simple dos and don'ts.

Table 3.1: Dos and don'ts to achieve data quality

Dos	Don'ts
Do start measuring for rapid baseline visibility	Don't force your teams to refactor or standardize their processes
Do ask teams to validate the metrics and fix any glaring data hygiene issues, e.g., closing out tickets when they are done	Don't precondition visibility on data hygiene
Do offer a variety of standard processes in cases where it would drastically simplify measurement	Don't mandate a single way of working for everyone

In the next sections, we'll explain how Faros AI handles sophisticated normalization for common scenarios without requiring upfront standardization. This includes:

- Multiple data sources for the same data
- Same tool, different workflows
- Same tool, different usage
- Different goals for different teams

- Basic data validation

Multiple Data Sources for the Same Data

It is quite common for different teams or sub-orgs to use different tools. For example, one team might manage their tasks in Jira, another uses Asana, and a third uses GitLab. Another example is a company with multiple instances of the same tool.

Normalization is very simple in these cases. The Faros AI connectors normalize the data upon ingestion, automatically mapping corresponding data types to the right place in our canonical schema.

Same Tool, Different Workflows

Another common scenario is projects within a single tool, like Jira, that use different workflows, as expressed by statuses. Faros automatically deals with status transitions and provides the desired breakdowns based on the level of analysis:

- Each team's particular workflow will be represented in its metrics, so team members can understand their bottlenecks, learn, and affect change where needed.
- At the leadership level, where we're zoomed out to team-of-teams or much larger groups, metrics will be abstracted to the common statuses of To Do, In Progress, and Done. This is sufficient to see the bottom line metrics leaders care about, like task cycle time and amount of work in progress.

Same Tool, Different Usage

Every team in your organization might be using Jira, but they're using it very differently. Normalization is required to report effectively across this variance in tool usage.

The Faros AI approach is to be compatible with how people work today, especially at the very beginning of your program. To that end, the data normalization can be handled in a couple of ways:

1. **By building conditions into the chart queries.** For example, let's imagine you want to look at all high-priority unassigned issues. One team may use P0 and P1, while another uses Critical and High. A custom query can bake these different definitions into the chart.
2. **By using the platform's data transform capabilities.** For example, one group uses epics to track initiatives, while another group uses tags on tasks, and a third group uses a custom issue type. Faros AI transforms the data to the initiative portion of our schema, so you can then query all the initiatives in a single way.

At some point, if the maintenance of the queries or transforms becomes too complex and error-prone, Faros AI recommends introducing a few standard options. You don't force everyone

to comply to the same behavior, rather to select one of a handful of approved ways of doing things. This should cover the majority of team preferences while keeping the in-tool configurations manageable.

Different Goals for Different Teams

Let's face it: Good and bad are relative.

Consider one product under active development and another product that is in maintenance mode. While you may want to measure the same things for these teams — for example, throughput — they will have different definitions of good. "Good" is also relative to a baseline, and their starting points may be wildly different.

The Faros AI approach is to make it easy for every role to easily understand how teams are performing relative to contextual goals.

- **Teams can customize their thresholds for great, good, medium, and bad.** These custom thresholds will be utilized for their personalized dashboards featuring team-level metrics and insights.
- **Leaders will get a bird's-eye view at the organizational level that takes all the personalized thresholds into account and visually identifies hotspots**. It will also call out areas of improvement or decline.

Note: Popular frameworks like DORA publish annual benchmarks, but the way the metric is defined might not be applicable to how you work. For example, Deployment Frequency measures how often you deploy code changes to production. If your organization has a major product release four times a year, strict adherence to that definition won't give you the insight you seek. In this example, Faros AI recommends measuring deployment frequency to your pre-prod environments.

Basic Data Validation

As Faros AI begins to ingest and normalize data, it will identify gaps in data collection and mapping. Through troubleshooting and cleanup, you can address and fix these errors. In addition, charts can be tweaked, for example, to use median instead of average.

During this phase, you may also discover places where processes are not being followed internally and need to address the issue with the relevant teams.

- **Faros AI produces a report that shows data gaps**, including boards without an owner and teams without data.

- **Faros AI highlights anomalies and outliers,** primarily found in human-curated data, like tasks that have been open for hundreds of days or tasks that moved directly from ToDo to Done.

Once your data has been validated, you've paved the way to the next stage of analyzing it.

Chapter 4: **Analyzing the Data**

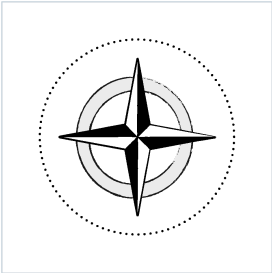
Once your data has been ingested and normalized, your dashboards and scorecards begin to populate. Now it's time to start understanding the data.

The beauty of centralizing and integrating your data is that by making those connections, you'll be able to uncover the contributing factors to current performance and understand what to improve.

In this chapter, we'll discuss how to start comprehending the data on day one and as you deepen your examination. In Chapter 5, the final chapter of this handbook, we'll discuss how to operationalize the data to become a data-driven organization.

In this chapter, we'll answer these questions:

- How do we begin to understand what our metrics are saying?
- How do we uncover trends and extract clear insights to drive improvement?
- How do we optimize processes and reduce bottlenecks?



Guiding Principle: Take a Role-Based Approach

Data is collected to serve two purposes:

1. **To answer questions you know to ask.** For example, "How are we doing on the five reliability KPIs we're doubling down on this year?" Out-of-the-box or customized dashboards will help you measure and track these things. You can also conduct ad-hoc analysis to answer new questions as they come up.

2. **To learn new things it would not occur to you to ask**. For example, "In the Mobile and Data Platform groups, the ratio of engineering managers to developers is higher than industry benchmarks, indicating an opportunity to rebalance resources." These insights are

provided by Faros AI curated analytics and its Lighthouse AI statistical analysis and machine-learning features.

Because a person's role and seniority in the organization determine both their questions and what they want to be told, the guiding principle for analyzing data is to apply a role-based lens.

Figure 4.1: Applying a role-based lens to day one analysis

The first thing to do is get a sense of overall health for the performance dimensions you care about, confirm your gut feelings, and identify hotspots. This is what we call "day one analysis." The table below lists the day-one metrics and analysis dimensions recommended for each role.

Table 4.0 Day-one engineering productivity analysis by role

Role	Analysis Dimensions	Day-One Questions to Answer
Senior Engineering Leader	Top-down for my organization, split by sub-org and teams	• KPIs, overall health, and hotspots • Key initiative progress • Cost vs. impact • Resource allocation
Line Manager	Bottom-up for my team, split by service, repo, and IC	• Baseline velocity and quality • Performance trends • Bottlenecks and dependencies
Functional Leaders (e.g., Product, TPM, PPM)	Horizontal for my domain, split by teams, epics, initiatives	• On-time delivery • Process efficiency • Planning efficacy • Coaching efficacy • Bottlenecks and dependencies

Role	Analysis Dimensions	Day-One Questions to Answer
Functional Leaders (e.g., Platform Engineering, DevEx)	Horizontal for my domain, split by teams, repos, and pipelines	• Tool utilization and impact • Delivery velocity and bottlenecks • Developer satisfaction
Functional Leaders (e.g., QA, Security)	Horizontal for my domain, split by teams, repos, or services	• KPIs, SLAs, SLOs • Cost vs. impact
Initiative Leaders (e.g., Migration, Modernization, AI)	Horizontal for my initiative, split by teams	• Resource utilization • Budget planned vs. actuals • Risk
HR Leaders (Team of Teams)	Top-down for my employees, split by geo, employment type, tenure, manager	• Engagement and retention health • Skill and composition health • Hotspots

After you've gotten the initial read from the data (and triggered some data cleanup, if warranted), it's time to go deeper. In this section, we'll provide guidance on how to:

- Quickly gauge the current state with benchmarks
- Counteract data overload with visualization
- Counteract data overload with AI
- Validate and contextualize findings
- Perform ad-hoc, impromptu analysis

Quickly Gauging the Current State with Benchmarks

Organizations like to reference industry benchmarks to gain a better perspective on their comparative strengths and weaknesses. These popular benchmarks are often born of extensive research that ties high performance to better financial performance, which can help:

- **Set priorities:** Where do we start?
- **Set goals:** What should we aim for?
- **Justify investments:** How do we incrementally become world-class?

Faros AI incorporates many software engineering industry benchmarks and best practices for velocity, quality, reliability, predictability, security, and organizational composition to help you quickly evaluate your situation. These metrics are provided at every level of analysis.

Benchmarks include but are not limited to the DORA 4 (lead time, deployment frequency, change failure rate, and mean time to restore), cycle times, velocity, say/do ratios, planned vs. unplanned

work, AI coding assistant impact, and staffing ratios. Use these benchmarks to understand your relative standing and surface performance areas to focus on.

Counteracting Data Overload with Visualization

Senior leaders typically benefit from effective summarization by a single powerful visualization. The Faros AI scorecard consolidates the top 5–10 KPIs the organization has chosen as the most impactful leading and lagging metrics. It highlights which areas of the organization are performing well and which require attention. From there, it's easy to drill down into sub-orgs and more detailed views, if required.

Counteracting Data Overload with AI

A lot of data can be both a blessing and a curse. Engineering leaders don't have the time or capacity to spend hours sifting through dashboards and digging into the details. Like any leader, they prefer the key takeaways and interesting insights delivered to them.

Faros AI is like a group of expert data analysts to augment your staff and provide a daily or weekly briefing. It utilizes AI, layer upon layer, from the simple to the complex, to **shorten the time between identifying an issue and taking corrective action**.

Lighthouse AI, the built-in AI engine of the Faros AI platform, applies statistical analysis and machine learning to pinpoint problem areas in specific sub-orgs, repos, applications, or stages of the SDLC. It automates very difficult and time-consuming analysis if done manually.

Lighthouse AI uses a proprietary machine-learning workflow to analyze key engineering metrics against 250+ factors that can impact them. It then presents personalized team-tailored insights into what's inhibiting or improving performance. It also leverages LLMs to clearly summarize and explain the findings and recommend solutions.

Here are the different ways Lighthouse AI works.

Table 4.1: Lighthouse AI capabilities to counteract data overload

Individual Metrics	Rich Dashboards	Conversational Chat
Explains the metric: How is this metric computed?	Explains the dashboard: What is this dashboard about?	Helps query data: How can I measure ABC?
Summarizes the data and key takeaways: What are the findings?	Summarizes the data and key takeaways: What are the issues?	
Alerts when a threshold is exceeded: When is it time to take action?	Recommends next steps: What should I do to address the issue?	
Identifies outlier teams: What is causing significant variations in performance?		

Individual Metrics	Rich Dashboards	Conversational Chat
Recommends next steps: What should I do to address the issue?		

Validating and Contextualizing Insights

Now that you've baselined your data, you've identified hotspots, bottlenecks, or areas of friction. You've also begun to uncover their contributing factors — with or without AI. Next, it's time to validate and contextualize the information.

No one knows your business like your developers and managers, so any statistical finding should be validated by the people involved.

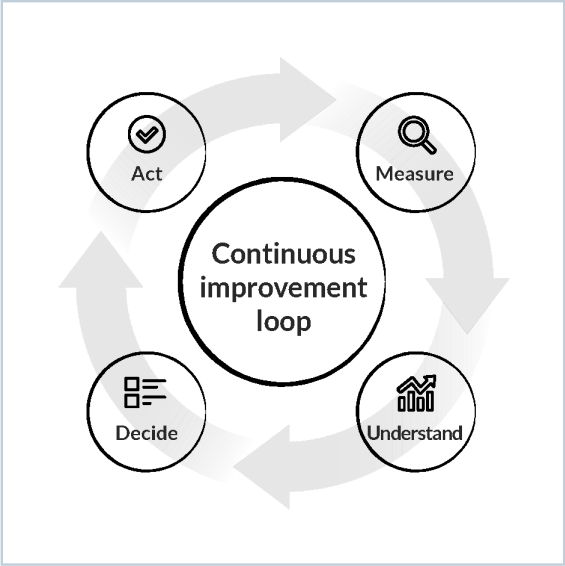
The Faros AI approach recommends the following:

1. **Measure:** Gather the data points indicative of an issue.
2. **Understand:** Deepen your understanding of the issue by adding context.
    a. **Talk to your team.** Speak to your team members to get their perspective. Learn what they think is happening and why. This is where you tie the telemetry to all your institutional knowledge and business context.
    b. **Consult developer survey data.** Developer surveys should ask developers what's causing the most friction in their daily work. If you're not yet importing your qualitative developer survey data into Faros to correlate it with quantitative data, consider doing that now. If not possible, manually review recent developer surveys to see what they say about the issue.
    c. **Confirm the priority.** How important and consequential is this issue? Arrive at a consensus on whether this issue is worth pursuing now.
3. **Decide: Formulate a hypothesis and customize visibility to accompany your analysis.**
    a. **Formulate a hypothesis**, time-box it, set a goal, and consider the different levers you have to take action.
    b. **Set up granular metrics for the workflow or process in question**, if required. For example, if the issue pertains to incident resolution, configure a flow in Faros AI that will allow you to examine every step of the process across tools and interactions like Detect > Create > Triage > Resolve > Restore. This will help you measure the impact of your changes throughout its cycles.
    c. **Create new charts or dashboards** that will enable you to examine the factors at play. For example, if your hypothesis is that incident resolution times are influenced by cross-geo delays, customize a chart to incorporate the geographical location of the team members involved.

   d. **Configure alerts or notifications** that will be helpful in tracking or addressing the issue.

4. **Act.**
   a. **Enact the change.** Designate someone to be focused on the issue and the change management process. They should enable the teams, monitor the impact, and share the learnings.
   b. **Monitor the impact.** After acting, keep your eyes on the metrics. Is the change helping you achieve your goal? Are there unintended consequences you need to address?

Figure 4.1 Validating and contextualizing insights with the continuous improvement loop



Performing Ad-Hoc, Impromptu Analysis

Looking at data frequently prompts additional questions, like:

- Why was March such an outlier month?
- Did the new process we implemented in June impact our performance?
- What does last year's holiday season tell us about this year's planning?

Now that you have data at your fingertips, you can answer those questions faster.

> *Last quarter, our SVP asked me for information and walked away, expecting it to take a couple of days. I was back in five minutes with all the data.*
>
> — Matt Runkle, Sr Director of Engineering, SmartBear

Faros AI recommends that the organization designate at least one data analyst for the engineering productivity initiative. This individual should be deeply familiar with your business, priorities, and ways of working. During the implementation and customization phase, Faros AI will train them on the Faros AI data model so that they develop the expertise to produce meaningful metrics and dashboards efficiently and independently.

In addition, platform end users can leverage wizards and natural language prompting to find answers. When relevant, Faros AI will suggest existing charts that meet the bill and guide the user to apply custom filters, aggregations, and visualizations to get the precise chart they want. When a custom metric is in order, a chat interface helps users learn how to build a custom query, including which tables to use, key information about the data, and a variety of tips and tricks.

Chapter 5: **Operationalizing the data to achieve impact**

Tools don't magically change your company. It's your use of them that does.

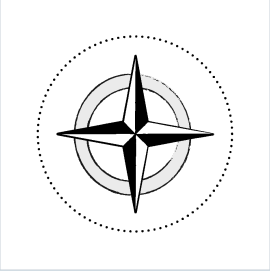
Implementing a software engineering intelligence platform is similar to implementing Salesforce. Salesforce doesn't magically increase sales. It's the use of Salesforce data in recurring cadences to inform decision-making and strategy that does.

The Head of Sales has to make a practice of reviewing key sales metrics and performance indicators in weekly meetings. Salespeople are expected to keep their data up to date, track their progress towards goals, and come to meetings prepared to discuss next steps and priorities. In QBRs, data from Salesforce is pulled up in support of the review and the proposed action plan.

Instilling similar behaviors enables engineering's transformation to data-driven engineering operations.

In this chapter, we'll answer these questions:

- How do we become a data-driven organization?
- How do we use our engineering data to continuously improve?
- How do we make sure we're investing in the right things and are resourced accordingly?



Guiding Principle: Incorporate Data in Your Recurring Cadences

Engineering is a big and important function, supported by recurring cadences designed to facilitate organizational learning and growth and to ensure objectives are met.

Prior to having a platform like Faros AI, these cadences were often conducted based on partial visibility, low-quality data, one-off spreadsheets, and gut feelings. All of these cadences benefit from the injection of golden, objective data.

> *The biggest benefit we see is that we no longer rely on gut feelings to set our action items. We now have a combined picture from all the tools we use and can do much more sophisticated analysis.... Our transition to data-driven retros has energized and motivated the team; they love seeing the impact of their efforts in the charts.*
>
> — Elad Kochavi, Engineering Manager, Riskified

For many organizations, change management will be necessary to modify existing meeting protocols and practices to now include the presentation and discussion of integrated data.

The table below illustrates the guiding principles for implementing the transition to data-driven engineering operation.

Table 5.1: Transitioning to data-driven engineering operations

1	2	3	4
Establish ownership and accountability for change	**Tailor visibility to support recurring cadences**	**Make resource allocation and decision approvals contingent on supporting data**	**Ensure every team is accountable for its data & continuous improvement**
Imbue an internal champion with the authority to institute new data-driven practices	Make it easy to run meetings and business processes with data by building customized dashboards for each cadence	Require data evidence for resource, prioritization, and HR-related decisions	Train teams on their metrics and how to ensure data quality and accuracy

In the following sections, we'll review how to inject data into your main recurring cadences through the five-pillar model of engineering operations.

Injecting Data into the Five Pillars of Engineering Operations

Whether a scaling startup or a mega-enterprise, world-class engineering organizations operate on the foundation of five essential pillars: Budgets, Talent, Productivity, Delivery, and Outcomes. These pillars ensure that operations are efficient, strategic, and aligned with the company's goals.

Figure 5.1: The five pillars of engineering operations

Each pillar is reinforced by specific recurring processes and cadences that facilitate sustained performance and growth. When these meetings are fueled with high-quality, evergreen data, decisions are made faster and more confidently.

As mentioned in Chapter 1, most organizations launch their programs with productivity in mind and then expand clockwise to the other pillars.

In the following sections, we describe the main recurring cadences in each pillar and the recommended metrics to review in each:

- Productivity cadences and metrics
- Delivery cadences and metrics
- Outcomes cadences and metrics
- Budgets cadences and metrics
- Talent cadences and metrics

Productivity Cadence Metrics

Platform Engineering, Developer Experience, and Architecture teams implement ongoing initiatives to modernize technology, optimize workflows, enhance collaboration, and remove bottlenecks.

Monthly operational reviews are structured to review key metrics, address challenges, and align on priorities. Project reviews focus on the progress and impact of foundational transformations like migration, modernization, compliance initiatives, and tool and technology implementations.

Table 5.2: Metrics for recurring cadences focused on productivity

Productivity Recurring Cadences	Metrics
Group-level operational review	On-time release success rates Release delays vs. target Feature rollout progress Recent service-level metrics Production incident review ("SEVs") Tool/process adoption
Product or team technical review	Analyzed for each component within the team's scope of ownership or from the product lens: Deployment frequency What shipped this quarter Activity volume by work type Velocity Service-level metrics (e.g. uptime, performance, production incidents, KTLO, number of production defects, vulnerabilities by severity) Qualitative feedback
Periodic developer experience survey review	Analysis of qualitative responses Correlation of qualitative responses to quantitative data Hotspot identification

Delivery Cadence Metrics

Delivery is most commonly managed through agile methodologies, with work segmented into short, iterative cycles such as sprints or development iterations. Regular stand-ups, sprint planning, and retros help ensure projects remain on track and deliverables meet quality standards. Monthly product and tech reviews provide a structured forum for assessing progress, addressing issues, and aligning priorities.

Table 5.3: Metrics for recurring cadences focused on delivery

Delivery Recurring Cadences	Metrics
Quarterly Planning	Previous quarter look-back to inform next quarter: Velocity, throughput, and quality metrics to adjust capacity estimations, priorities, and commitments

Delivery Recurring Cadences	Metrics
	Resource allocation across initiatives and work types Previous quarter's work distribution % completion of initiatives and analysis of outstanding work Cross-team dependencies and bottlenecks Adherence to targeted resource allocation (strategic/innovation, bug fixes, KTLO, tech debt)
Sprint Retros	Planning accuracy and predictability: Say/Do ratios Unplanned work percentage Developer satisfaction Task and PR Hygiene Backlog health WIP Progress vs. goals: % completion of epics and analysis of outstanding work
Initiative/project reviews	Initiative progress vs. plan and budgets Predicted delays and cost overruns Breakdown of remaining work by epic and team

Outcomes Cadence Metrics

Organizations operate on a quarterly cadence to set, review, and adjust their objectives and key results (OKRs). This process aligns teams with strategic goals, measures progress, and ensures accountability. Regular check-ins and quarterly reviews facilitate continuous alignment and goal achievement. Cross-functional QBRs provide a comprehensive evaluation of the quarter's performance against the set OKRs.

Table 5.4: Metrics for recurring cadences focused on outcomes

Outcome Cadences	Metrics
C-Suite and Board Reporting	Engineering as a % of revenue Revenue per engineer
Quarterly reviews (start, mid, and end of quarter)	*All of the above, plus:* Initiative progress: Initiative progress vs. goals and risks Remaining work per initiative Risk mitigation strategies

Outcome Cadences	Metrics
	Cost vs. impact Business metrics: Growth metrics Customer retention metrics QoQ North Star metrics (adoption, usage, revenue, retention, security, reliability, CSAT) Engineering as a % of revenue Revenue per engineer Roadmap metrics: Delivery update (completed, in progress, up-next) Committed work by investment category Delivered work by investment category Resource allocation across the portfolio Portfolio/Product KPIs: % delivered vs. committed, and cause/reason Initiative progress vs. goals Productivity vs. goals (velocity, throughput) Quality vs. goals (number of critical defects, open and closed bugs by team, code coverage) Availability SLAs (e.g., closed bugs SLA, open bugs SLA, % of bugs solved out of SLA) Support SLAs (MTTR, code coverage, CSAT) Security SLAs Bottlenecks and dependencies

> *In our QBRs, metrics in Faros AI help map engineering's work to business value. The excellence with which our engineering teams deliver and operate can be tied directly to the lagging results, like helping the business acquire, retain, upsell, or increase customer satisfaction.*
>
> — Shai Peretz, SVP Engineering, Riskified

Budget Cadence Metrics

Engineering organizations begin their annual strategic planning towards the end of the fiscal year. This process involves forecasting financial needs, allocating resources, and setting financial goals for the upcoming year. Budget reviews and adjustments are made throughout the year to ensure alignment with evolving priorities and market conditions.

Table 5.5: Metrics for recurring cadences focused on budgets and staffing

Budget Recurring Cadences	Metrics
Annual planning and quarterly review	Impact: Engineering as a percentage of revenue Revenue per engineer ROI by engineering initiative Resource allocation vs. revenue Productivity: Productivity vs. benchmarks Productivity per location Efficiency: Organization composition by role Management overhead vs. benchmarks Functional staffing ratios vs. benchmarks
Global sourcing review and planning	Productivity per dollar spent per location Productivity per employment type Productivity per contract type
Vendor and subcontractor contract negotiations and renewals	Productivity per dollar spent per subcontractor Vendor costs and ROI
Periodic accounting events	Capitalized software development costs per period

Talent Cadence Metrics

Talent reviews and performance evaluations are conducted twice a year to assess employee performance, identify high-potential talent, plan for career development, and identify skill gaps. Compensation reviews, which are typically implemented in March or April, ensure that employee remuneration reflects performance and market standards, thus attracting and retaining top talent.

Incorporating objective data in these processes reduces subjectivity, speeds up preparation, and increases confidence in decision-making. It also ensures feedback is evidence-based, leading to clear and actionable development goals.

Table 5.6: Metrics for recurring cadences focused on talent management

Talent Recurring Cadences	Metrics
Performance reviews	Impact: PRs authored and tasks completed with respect to size, complexity, and impact (over time, by work type, by epic) PR and task cycle times relative to team/peers

Talent Recurring Cadences	Metrics
	Quality: PR quality (PR size, code quality analysis, review time, number of comments, number of reverts) Collaboration: Number of PR reviews completed Number of authors reviewed Number of reviewers Number of bugs and incidents resolved Calibration: Comparison to similar cohorts by role, seniority, and tenure
Talent reviews and workforce planning	Number of engineers (by level, by job profile) Functional staffing ratios vs. benchmarks Team composition outliers vs. benchmarks (by level, by job profile) Open headcount (by level, by job profile) Attrition rate Number of at-risk attrition employees Technology and coding language usage Developer CSAT
Organization structuring	Productivity per location Productivity per employment type Productivity per dollar spent Engineering overhead ratio Cross-team dependencies

Handbook Summary

Question	Guiding Principle	What To Do Next
What should we measure?	**Identify what matters to you.** Tailor your program to your business context, otherwise there will be no buy-in.	Identify your **goals** based on your company's stage. Understand how your **operating model** dictates key analysis dimensions. Take care to preserve your **engineering culture** in your approach to metrics.
Which data sources should we collect?	**Begin with the basics and advance.** Take a step-wise approach to achieve quick wins while gradually completing the picture.	**Baseline** with task, PR, and org chart data. **Blend** in developer surveys and collaboration tools. **Expand** to form a holistic view of quality, reliability, and security. **Align** to corporate objectives with business outcome tracking.
How can we normalize and validate the data?	**Don't let the perfect be the enemy of good.** Data quality is a result of visibility, not a pre-condition. Start measuring to create high-level visibility.	Metrics will highlight the inconsistencies that teams should address without enforcing wholesale standardization. If leaders demonstrate they care, teams will address the data gaps, anomalies, and outliers identified by Faros AI.
How do we begin to understand our current state?	**Take a role-based approach.** Get a sense of overall health for the performance dimensions you care about, validate gut feelings, and identify hotspots.	**Gauge** the current state with benchmarks. **Counteract data overload** with visualizations and AI. **Validate and contextualize** findings. **Perform ad-hoc, impromptu analysis** to answer emerging questions.
How do we operationalize the data to achieve impact?	**Change management is essential to becoming a data-led organization.** Inject data to your recurring cadences and decision processes across the five core pillars of engineering operations.	**Productivity:** Monthly operational reviews and project reviews. **Delivery:** Quarterly planning, sprint retros, and initiative reviews. **Outcomes:** C-Suite and board reporting, quarterly reviews, and midpoint check-ins. **Budgets:** Annual planning, quarterly reviews, vendor contract negotiations, global sourcing strategic planning, and periodic accounting/compliance events. **Talent:** Performance reviews, workforce planning, organization structuring.

Join our community