embrace

Rethinking KPIs:

How engineering can own business outcomes

Intro

This guide is designed for engineering leaders to address a persistent challenge: the disconnect between how businesses measure success and how engineering teams monitor systems.

Businesses focus on outcomes like revenue, daily active users (DAU), and churn, while engineering teams typically track system-centric metrics such as uptime, crashes, and response times. Without a shared language or framework to bridge this gap, critical problems often remain undetected until key performance indicators (KPIs) have already been negatively impacted.

This guide introduces a framework to help engineering teams translate business KPIs into clear, actionable technical insights. The framework is anchored on user-focused observability, a practice that goes beyond traditional observability to prioritize real user impact. Engineers, therefore, can tactically adapt these tools and data-driven techniques to proactively solve problems related to business KPIs and maintain strategic alignment with commercial teams.

Want to learn more about how you can operationalize KPIs?
Get the guide.

01

THE KPI DISCONNECT: BUSINESS GOALS VS. TECHNICAL SIGNALS

A fundamental disconnect often keeps organizations in a steady state of dysfunction. Executives and commercial teams design OKRs or KPIs that have broad, organization-wide impact, such as customer churn or revenue targets. Meanwhile, engineering teams maintain their own sets of performance standards based on traditional, very technical metrics. These are things like system uptime, latency, crash-free rates, etc.

These two distinct sets of metrics rarely connect directly. This is unproductive and inefficient because, in reality, both sets of metrics are deeply connected and should work in tandem to achieve the same goal.

One of the problems preventing this from happening is that purely business-level KPIs are very abstract and all-encompassing; it is often difficult to isolate exactly what factors influence a metric or which team is responsible. On the flip side, technical KPIs are often too “in the weeds” and hyper-specific. In some cases, an engineering KPI might be “technically” hitting its mark, but it’s too isolated from other systems and functions to be truly indicative of whether core product functionality is working like it’s meant to, from the perspective of customers.

In the grander scheme, this disconnect between business and technical KPIs often leads to some common challenges:

- **Business KPIs become lagging indicators:** Critical product issues are only detected after negative impacts like revenue loss or erosion of customer trust have already occurred.
- **Prioritization struggles:** Technical teams often find it difficult to prioritize issues effectively because they lack a clear understanding of the specific business impact associated with each technical problem.

For example, a business KPI might be "orders per minute.” If it suddenly drops, traditional infrastructure dashboards might not be helpful as they may not show any obvious technical anomalies, leaving teams blind to the root cause and delaying identification and resolution.

In the “orders per minute” example, frontend issues not picked up by traditional dashboards might look like:

- A mobile carrier introduces packet loss or throttling in a specific region, leading to API calls intermittently failing or taking 5–10 seconds longer
- A CDN node serving assets (images, CSS, JS bundles) in Europe has a misconfiguration, so those assets load extremely slowly for EU users
- A payment gateway SDK (e.g., Stripe, PayPal) times out for mobile Safari users, preventing them from completing a purchase

This disconnect can prevent engineering from acting proactively to protect business outcomes, while leaving other stakeholders struggling to get ahead of issues that negatively impact their commercial KPIs.

02

WHY EVERY BUSINESS KPI IS BUILT ON USER ACTIONS

Breaking this frustrating cycle requires a different approach to software KPIs, one that:

- (1) puts end user impact, rather than specific technical elements, front and center, and
- (2) sets up translational layers between business outcomes and the technology that drives them

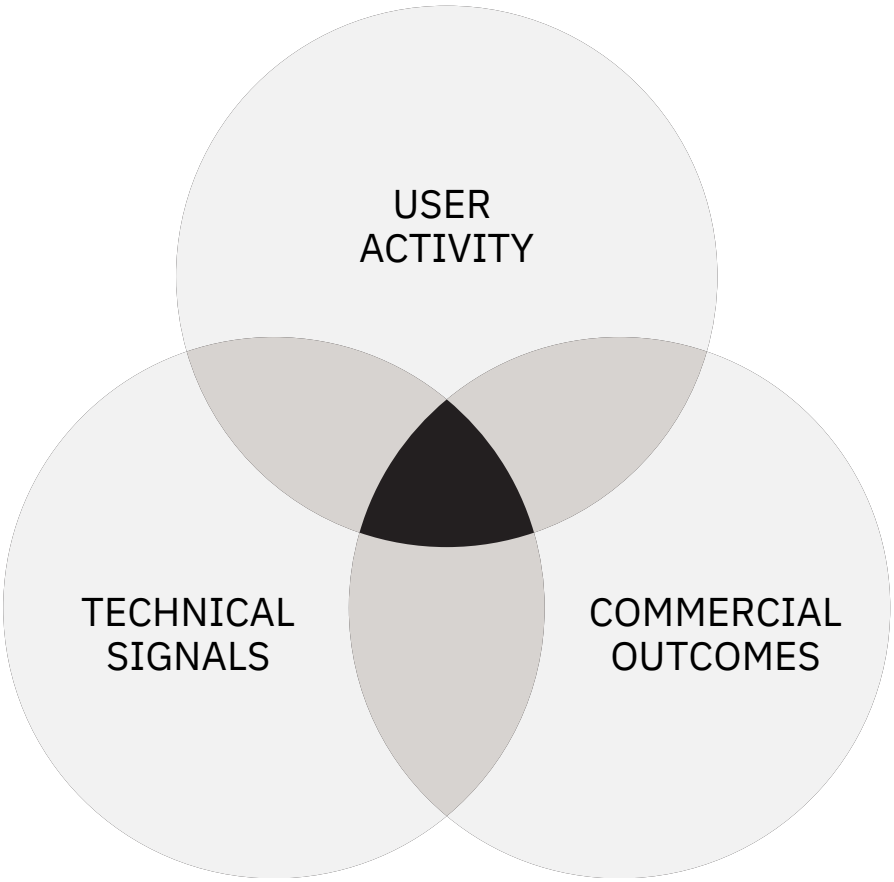
Fundamentally, all business outcomes are the direct result of individual user interactions. Every major business KPI is an aggregation of smaller, discrete user actions.

Consider the "orders per minute" KPI:

- It is composed of user actions, such as searching for a product, adding items to a cart, and completing the checkout process.
- Each of these user actions relies on various technical dependencies, including frontend rendering, API latency, and the uptime of payment services.

This means that even micro-level failures — such as a degraded checkout API performance on specific devices — can cascade upwards and directly lead to declines in overall business KPIs. Therefore, proactively protecting KPIs necessitates tracking user flows and their underlying technical dependencies end-to-end.

To think of this from the opposite perspective, prioritizing engineering reliability work also requires understanding how certain technical functions are directly related to big-picture, customer-impacting outcomes.



03

IMPLEMENTING THE TRANSLATIONAL LAYER

Many organizations struggle with this gap between business KPIs and technical reliability metrics. Bridging this requires a translational layer that breaks down business KPIs into more manageable components, while simultaneously contextualizing technical data.

The first tool you might reach for to conceptualize this layer is product analytics.

Product analytics tools (such as Mixpanel, Amplitude, and Google Analytics) are extremely useful. You would be hard pressed to find an organization that does not use this type of data on a regular basis across Product, Engineering, Design, and Marketing teams.

These tools are excellent at telling teams what is happening, providing valuable insights into user behavior and trends. You might glean high-level data points from product analytics like:

- Conversion rate has dropped 12% this week
- Users are abandoning carts at the payment screen
- iOS 18 users are engaging less with push notifications

WHERE PRODUCT ANALYTICS FALL SHORT

While these insights are highly valuable, they stop short of providing the technical visibility needed to answer the next critical question: Why?

Product analytics primarily focuses on user actions and behavioral trends. Critically, they cannot reveal the technical dependencies behind those actions. They are great at telling you what is going on, but are extremely limited in giving you real, directional insight into why it's happening.

For example, a product analytics tool can effectively show that checkout completions declined, but it cannot tell you that the payment API is intermittently failing for users on 3G networks.

The result:
Without this crucial link between behavioral data and technical root causes, teams often end up playing a guessing game. Product managers observe a problem in the user funnel, but engineering teams lack the precise context required to reproduce or resolve it efficiently. If this problem is affecting a critical function like log-in or payments, its effects inevitably cascade all the way to business outcomes.

Your KPI is now suffering and, although you can see what features are most likely responsible, you're not all that much closer to resolving the root cause.

This gap between "business visibility" (the WHAT) and "technical observability" (the WHY) inevitably slows down problem resolution and can lead to friction between product, engineering, and commercial teams.

WHERE USER-FOCUSED OBSERVABILITY COMES IN

User-focused observability is a practice of monitoring software systems with an emphasis on how the reliability and functionality of these systems is perceived by the end user who is trying to accomplish something on your site or app. This is in contrast to traditional observability, which focuses on tracking internal components, like servers and databases.

User-focused observability complements product analytics by directly bridging this gap between the technical and behavioral.

This looks like:

- Linking what is happening (e.g., a decline in conversions) with why it’s happening (e.g., an API timeout for a specific device or operating system).
- Providing end-to-end context by mapping user actions (like "add to cart") directly to the technical dependencies that power them.
- Equipping engineering with precise failure data (including 100% session capture, detailed error traces, and network spans) so issues can be reproduced and fixed quickly.

Traditional observability tools, which primarily focus on logs, metrics, and traces, often provide a fragmented and system-centric view. While valuable, they typically do not inherently connect technical data back to business outcomes.

In contrast, user-focused observability fundamentally shifts this perspective by starting with the end-user journey and then connecting it to the various technical components. This approach offers several critical benefits:

Outcome-based metrics: Engineering teams can establish alerts based on the health of specific KPIs and user flows, rather than solely on raw error rates.

Granular technical mapping: It allows for the precise pinpointing of which device, geographical region, or specific dependency is experiencing a failure.

Business prioritization: Teams can effectively rank and prioritize engineering resources based on their direct impact on revenue or customer experience, moving beyond mere technical severity codes.

This method essentially creates a "Business KPI → User Actions → Technical Dependencies" funnel, providing a clear path from business outcomes to technical realities.

REAL-WORLD USE CASE

Consider the case of a popular e-commerce application where "orders per minute" suddenly experienced a dip. Despite this critical business impact, legacy monitoring systems showed no major outages or anomalies, leaving the engineering and product teams without any clear indication of the problem.

However, by implementing user-focused observability, the real cause was quickly identified: There was a timeout issue with a specific payment processor, but only for iOS users running the latest operating system version. Because the issue was precisely pinpointed and understood in terms of its user and business impact, the teams were able to resolve it rapidly. This proactive approach restored the KPI's health and prevented the issue from escalating into significant revenue loss and customer churn.

04

FRAMEWORK: MAPPING BUSINESS KPIS TO TECHNICAL METRICS

To help engineering leaders apply this concept, here's a practical framework to break down KPIs into their constituent user actions and technical dependencies.

Business KPI	User action(s)	Technical dependencies	Failure impact
Orders per minute	Add to cart, checkout	API latency, payment gateway, frontend	Lost revenue
DAU/MAU	Login, session start	Auth service, SDK initialization	Decline in active users
Retention rate	Session duration, feature use	Crash-free sessions, latency	Increased churn
Customer satisfaction	Smooth navigation, load times	Rendering performance, third-party SDKs	Lower NPS / reviews

The key is to have a tangible way to link high-level business goals to the specific technical components and the user interactions that underpin them.

05

ACTION PLAN: HOW ENGINEERING LEADERS CAN PROTECT KPIS

For engineering leaders to effectively protect business KPIs and align technical work with commercial outcomes, it's important to create a clear action plan that integrates user-focused observability.

This might include steps like:

- Align with product and business leaders on which KPIs matter most to the organization.
- Clearly define the specific user flows and critical transactions that directly power those identified KPIs.
- Implement observability solutions that are capable of tracking these critical user flows end-to-end across the entire technology stack.
- Incorporate user outcome metrics into traditional observability workflows.
- Continuously map technical signals (such as latency, errors, and crashes) directly to their corresponding business impact.

By adopting this strategic approach, observability transforms from a mere reactive troubleshooting tool into a proactive business enabler. And product analytics also becomes a more powerful directional indicator in tandem.

This empowers the engineering team to drive and protect strategic outcomes, working more cohesively with commercial stakeholders.

CONCLUSION: PROTECTING KPIS STARTS WITH THE USER

In summary, business KPIs are ultimately driven by user actions, whereas traditional observability approaches focus on measuring system health. This disconnect causes friction for engineering teams who are expected to prioritize efforts based on their business impact.

By translating business metrics into technical clarity, engineering leaders can align efforts towards driving business goals, such as:

- The ability to detect risks early, often before KPIs experience any significant dip.
- The power to prioritize fixes based on their direct revenue or customer impact.
- The means to align engineering work seamlessly with overarching strategic business outcomes.

To learn more about how leading teams are operationalizing this KPI-to-technical-signal framework, check out our follow-up KPI translation toolkit or contact us for a KPI audit.

Want to learn more about how you can operationalize KPIs?
Get the guide.

Embrace is the only user-focused observability platform that ties technical performance to end-user impact. Powered by OpenTelemetry, Embrace provides real user monitoring (RUM) across mobile and web, giving engineering teams the visibility and context they need to resolve issues faster, optimize performance, and deliver exceptional digital experiences.

Contact

1901 Avenue of the Stars #200, Los Angeles, CA 90067

(310) 461-1310

contact@embrace.io

embrace.io

Learn more at embrace.io or follow Embrace on LinkedIn, YouTube, or X.