TEXT ONLY

DevEx What Actually Drives Productivity

THE DEVELOPER-CENTRIC APPROACH TO MEASURING AND IMPROVING PRODUCTIVITY

ABI NODA, DX

MARGARET-ANNE STOREY, UNIVERSITY OF VICTORIA

NICOLE FORSGREN, MICROSOFT RESEARCH

MICHAELA GREILER, DX

Engineering leaders have long sought to improve the productivity of their developers, but knowing how to measure or even define developer productivity has remained elusive.[5] Past approaches, such as measuring the output of developers or the time it takes them to complete tasks, have failed to account for the complex and diverse activities that developers perform. Thus, the question remains: What should leaders measure and focus on to improve developer productivity?

Today, many organizations aiming to improve developer productivity are finding that a new developer-centric approach focused on developer experience (also known as DevEx) unlocks valuable insights and opportunities. Gartner Inc. reports that 78 percent of surveyed organizations have a formal DevEx initiative either established or planned.[7] Accordingly, there has been a continued rise in dedicated DevEx and platform teams being formed to help improve developer experience within their organizations.

Developer experience focuses on the lived experience of developers and the points of friction they encounter in their everyday work. In addition to improving productivity, DevEx drives business performance through increased efficiency, product quality, and employee retention. A 2020 McKinsey study found that companies with better work environments for their developers achieved revenue growth four to five times greater than that of their competitors.[13]

This paper provides a practical framework for understanding DevEx, and presents a measurement framework that combines feedback from developers with data about the engineering systems they interact with. These two frameworks provide leaders with clear, actionable insights into what to measure and where to focus in order to improve developer productivity.

WHAT IS DEVEX?

Developer experience encompasses how developers feel about, think about, and value their work.[9] In prior research, we identified more than 25 sociotechnical factors that affect DevEx. For example, interruptions, unrealistic deadlines, and friction in development tools negatively affect DevEx, while having clear tasks, well-organized code, and pain-free releases improve it.

A common misconception is that DevEx is primarily affected by tools. Our research, however, shows that human factors such as having clear goals for projects and feeling psychologically safe on a team have a substantial impact on developers' performance.[9] Improving DevEx increases not only productivity, but also satisfaction,

Points of friction can adversely affect the developer experience at all organizational levels.

engagement, and employee retention.[9]

Points of friction can adversely affect the developer experience at all organizational levels. Foundational issues such as excessive build times and difficult infrastructure often crosscut an organization, requiring attention from leaders or dedicated teams. More local concerns, by contrast, such as minimizing interruptions or clearly defining work, may require specific improvement efforts tailored to the needs of teams or individuals.

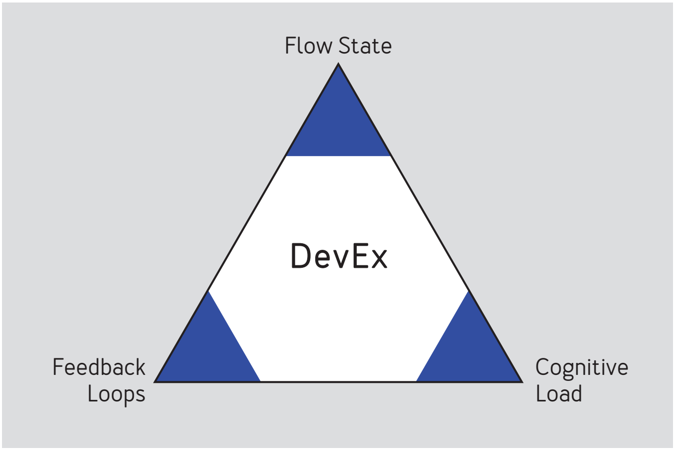
An individual developer's experience is highly personal and contextually dependent. The developer's team, role and responsibilities, and seniority all impact their lived experience. DevEx is best understood through careful attention to specific personas, as well as general organizational patterns. Specific approaches are recommended later in this article.

THE THREE DIMENSIONS OF DEVEX

Our framework distills developer experience to its three core dimensions: feedback loops, cognitive load, and flow state (illustrated in figure 1). These dimensions emerged from real-world application of our prior research, which identified 25 sociotechnical factors affecting DevEx.[9] These three key dimensions crosscut those 25 factors, providing a practical model for understanding DevEx.

Taken together, feedback loops, cognitive load, and flow state encapsulate the full range of friction types encountered by developers. Although DevEx is complex and nuanced, teams and organizations can take steps toward improvement by focusing on these three key areas.

FIGURE 1: **THREE CORE DIMENSIONS OF DEVELOPER EXPERIENCE**



Feedback loops

Software organizations commonly look for ways to optimize their value stream by reducing or eliminating delays in software delivery. This allows faster feedback and learning about what is being built, which in turn allows for more rapid course correction. Studies have consistently shown that organizations deploying more frequently and maintaining shorter lead times are twice as likely to exceed performance goals as their competitors.[3]

Shortening *feedback loops*—the speed and quality of responses to actions performed—is equally important to improving DevEx. A typical developer's day consists of numerous tasks relying on feedback from both tools and people. For example, developers may spend significant time waiting for code to recompile or tests to run. Later, they may wait for approval from code reviewers, which blocks their ability to make progress.

Fast feedback loops allow developers to complete their work quickly with minimal friction. Slow feedback loops, by contrast, interrupt the development process, leading to frustration and delays as developers wait or decide to switch tasks. Slow feedback loops cause additional interruptions when feedback from systems (such as an integrated test run) or people (such as review notes) is returned and requires immediate attention. To improve DevEx, organizations should shorten feedback loops by identifying areas where development tools can be accelerated (e.g., build and test processes, development environment setup) or human hand-off processes improved. Organizational structures should also be optimized to promote streamlined team interactions that minimize delay.

Organizations should shorten feedback loops by identifying areas where development tools can be accelerated or human hand-off processes improved.

Cognitive load

Software development is inherently complex, and the ever-growing number of tools and technologies is further adding to the cognitive load faced by developers. *Cognitive load* encompasses the amount of mental processing required for a developer to perform a task.[14] For example, cognitive load typically increases for a developer working on an inherently difficult or complex task, or learning to understand an unfamiliar development framework. Cognitive load also varies according to how external information is presented and increases when mental processing is required for translating information into longer-term domain knowledge and models.

Cognitive load impedes developers' most important responsibility: delivering value to customers. When cognitive load remains high as a result of problems such as poorly documented code or systems, developers need to devote extra time and effort to complete tasks and avoid mistakes.

To improve developer experience, teams and organizations should aim to reduce cognitive load by finding ways to eliminate unnecessary hurdles in the development process. Emphasis should be placed on creating well-organized code and documentation both to facilitate developers' understanding of the systems they work with and to reduce the amount of context or task switching required to complete their projects. Dedicated DevEx and platform teams should provide easy-to-use, self-service tools to help developers streamline the steps for development and release.

Flow state

Developers often speak of "getting into the flow" or "being in the zone." Such statements colloquially describe the concept of *flow state*, a mental state in which a person performing an activity is fully immersed in a feeling of energized focus, full involvement, and enjoyment.[1]

Frequent experiences of flow state at work lead to higher productivity, innovation, and employee development.[2] Similarly, studies have shown that developers who enjoy their work perform better and produce higher-quality products.[8] Interruptions and delays—which relate to the *feedback loops* dimension—are important factors that hinder a developer's ability to experience flow state.[10] Other factors include maintaining autonomy over work structure, having clear team and project goals, and engaging in stimulating and challenging tasks.[12]

To improve DevEx, teams and organizations should focus on creating the optimal conditions for flow state. Disruptions should be minimized by clustering meetings, avoiding unplanned work, and batching help requests. Leaders should also recognize that flow state depends on creating positive team cultures that give developers autonomy and opportunities to work on fulfilling challenges.

The key to measuring DevEx is to focus on developers and their lived experiences in delivering software.

MEASURING DEVEX

Organizations wanting to improve DevEx inevitably face the challenge of determining what and how to measure. Measuring DevEx is critical for identifying opportunities for improvement, detecting trends, and understanding the impact of investments.

The key to measuring DevEx is to focus on developers and their lived experiences in delivering software. This includes the performance of engineering systems, as well as the perceptions of developers who use and interact with those systems. Social factors such as collaboration and culture are important to capture as well.

Like developer productivity, developer experience cannot be boiled down to a single metric or dimension.[5] By recognizing and measuring DevEx with more than just a single dimension, teams and organizations can better understand how people and teams work, and can then make better decisions. This section presents a framework for determining what to measure, followed by recommendations for how to capture DevEx metrics in your organization.

What to measure

These three dimensions of DevEx provide a framework for identifying potential areas of measurement. Accurately measuring the full range of developer experience requires not only capturing developers' perceptions and workflows across these dimensions, but also overall KPIs (key performance indicators) or "North Star metrics" that capture the bigger picture. Table 1 Illustrates this approach, by pairing comprehensive KPI metrics with specific examples of perceptual and workflow measures

TABLE 1: **EXAMPLE DEVEX METRICS**

	FEEDBACK LOOPS	COGNITIVE LOAD	FLOW STATE
PERCEPTIONS *Human attitudes and opinions*	• Satisfaction with automated test speed and output • Satisfaction with time it takes to validate a local change • Satisfaction with time it takes to deploy a change to production	• Perceived complexity of codebase • Ease of debugging production systems • Ease of understanding documentation	• Perceived ability to focus and avoid interruptions • Satisfaction with clarity of task or project goals • Perceived disruptiveness of being on-call
WORKFLOWS *System and process behaviors*	• Time it takes to generate CI results • Code review turnaround time • Deployment lead time (time it takes to get a change released to production)	• Time it takes to get answers to technical questions • Manual steps required to deploy a change • Frequency of documentation improvements	• Number of blocks of time without meetings or interruptions • Frequency of unplanned tasks or requests • Frequency of incidents requiring team attention
KPIS *North star metrics*	• Overall perceived ease of delivering software • Employee engagement or satisfaction • Perceived productivity		

along the three dimensions. The next two sections elaborate on the differences between perceptual and workflow measures, and the importance of tracking overall KPIs.

PERCEPTUAL MEASURES VERSUS WORKFLOWS

Measuring DevEx requires capturing developers' perceptions—their attitudes, feelings, and opinions—in addition to objective data about engineering systems and processes. The "voice of the developer" provides direct signals across the three dimensions of DevEx through developers' self-reported perceptions and experiences. Objective data about systems and processes also provide insights into areas of friction, as well as recommendations for how to improve.

A comparative analysis of perceptual measures and workflows is necessary because neither alone can tell the full picture. For example, seemingly fast code review turnaround times may still *feel* disruptive to developers if code reviews regularly interrupt their work progress. As a counterexample, even if developers feel content with their build processes, using objective measures such as build time may reveal that feedback loops are slower and development workflows less streamlined than they could be.

Developers often have concerns about the misuse or misinterpretation of metrics by leaders. Measuring developers' perceptions can help leaders counterbalance metrics that are designed around a top-down view of developers' work. Staying focused on the lived experiences of developers ensures that leaders remain well-attuned to the true challenges faced by their developers.

THE IMPORTANCE OF KPIS

As previously mentioned, the three dimensions of developer experience are informed by 25 sociotechnical factors identified in our prior research. When capturing these dimensions, individual aspects should be measured to identify specific areas of the developer experience that can be acted upon and improved (see table 1 for examples).

By focusing on individual factors alone, however, organizations risk losing track of the bigger picture and investing in the wrong areas. It is important, therefore, that organizations also measure the higher-level KPIs for developer experience that serve as North Star metrics for DevEx initiatives. Well-designed KPIs should measure the outcomes that improvements to DevEx correlate with and seek to drive, including productivity, satisfaction, engagement, and retention.[9]

KPIs help organizations set strategic priorities and understand the full impact of their efforts. Reducing team meetings, for example, can improve the individual factors of developer experience relating to deep work. Reducing meetings can also make collaboration among coworkers more difficult, however, thereby hurting the KPI of overall satisfaction. In this example, attending to KPIs might steer teams toward strategies for improving time for deep work that do not simultaneously decrease satisfaction.

How to measure

DevEx should be measured by capturing developers' perceptions as well as their workflows in the various systems and processes that their work involves. Toward this end, organizations should collect data from both people and systems in order to gain full visibility into their software delivery processes.[4]

Surveys in particular are a crucial tool for measuring DevEx and capturing feedback from developers about points of friction in the software-delivery process. Survey data can be collected quickly (often within several weeks) and, when designed correctly, provide fast and accurate measurements to establish baselines and guide improvement efforts.[4]

To augment surveys, organizations should also collect data from systems. Getting end-to-end system data can be difficult, however, requiring instrumentation and normalization of data across disparate tools and teams.[11] Organizations can use surveys to capture self-reported information about systems (albeit with lower precision and frequency) in the absence of having fully instrumented systems.

Given the importance of surveys to the measurement process, the rest of this section outlines recommendations for organizations conducting surveys. These recommendations derive from our collective experience in partnering with hundreds of organizations to design and implement DevEx surveys.

DESIGN SURVEYS CAREFULLY

Poorly designed survey questions lead to inaccurate and unreliable results. At a minimum, survey questions should be based on well-defined constructs and be rigorously tested in interviews for consistent interpretation.[6] When possible, consult experts with significant experience in survey development.

Focusing only on aggregate results can lead to overlooking problems that affect small but important populations within the company.

BREAK DOWN RESULTS BY TEAM AND PERSONA

A common mistake made by organizational leaders is to focus on companywide results instead of data broken down by team and persona (e.g., role, tenure, seniority). As previously described, developer experience is highly contextual and can differ radically across teams or roles. Focusing only on aggregate results can lead to overlooking problems that affect small but important populations within the company, such as mobile developers.

COMPARE RESULTS AGAINST BENCHMARKS

Comparative analysis can help contextualize data and help drive action. For example, developer sentiment toward tech debt is commonly negative, making it difficult to identify problems or gauge their scale. Teams with lower sentiment scores than their peers and organizations with lower scores than their industry competitors, however, flag notable opportunities for improvement.

MIX IN TRANSACTIONAL SURVEYS

Transactional surveys capture feedback during specific touchpoints or interactions in the developer workflow. For example, platform teams can use transactional surveys to prompt developers for feedback when a specific error occurs during the installation of a CLI (command-line interface) tool. Transactional surveys can also augment data from periodic surveys by producing higher-frequency feedback and more granular insights.

AVOID SURVEY FATIGUE

Many organizations struggle to sustain high participation rates in surveys over time. A lack of follow-up action commonly causes developers to feel that repeatedly responding to surveys is not a worthwhile exercise. It is therefore critical that leaders and teams follow up on surveys. While a quarterly or semi-annual survey cadence is optimal for most organizations, for some, benefits may accrue from administering surveys more frequently.

REAL-WORLD EXAMPLES

Two companies, eBay and Pfizer, illustrate how attention to DevEx in their organizations is making a difference in productivity.

eBay: Accelerating developer velocity

As a global ecommerce leader, eBay is continually seeking new ways to make software delivery a competitive advantage for its business.

The company is undertaking a multiyear "Velocity Initiative" focused on improving developer productivity and experience. Improving feedback loops and cognitive load are key focus areas for eBay, and its internal platform and DevEx organization is responsible for gathering insights into problem areas in order to drive improvements.

The DevEx team measures developer experience through quarterly surveys that capture KPIs such as overall developer satisfaction and ease of development. Their surveys also capture developers' perceptions and workflows across different areas of their daily work. This data is augmented by realtime insights from engineering systems and focus group discussions to determine and implement specific improvements.

With these insights in hand, the DevEx team has led initiatives such as fixing tooling and documentation gaps across the organization, removing manual steps and processes for development and release, and streamlining cross-domain team interactions. To assist with rollout, the DevEx team closely partners with feature teams and conducts regular retrospectives, demos, and training sessions.

eBay's investments in DevEx continue to drive developer velocity and further accelerate progress toward establishing software delivery as a competitive advantage. In the past year, improvements enabled developers to release twice as frequently, and have resulted in a reduction of six times in deployment lead times.

Above all, eBay aims to ensure that developers are both excited to show up to work each day and have the tools they need to produce great customer outcomes.

Pfizer: Enabling breakthroughs at lightspeed

Since its breakthrough in developing a Covid-19 vaccine in nine months, Pfizer has embarked on a path toward improving DevEx in order to empower developers to deliver "breakthroughs at lightspeed."

Pfizer's engineering organization has grown nearly tenfold from 2018 to 2022 while simultaneously modernizing its software-delivery capabilities through open-source and cloud-native technologies. Today, Pfizer's DevEx team is focused on finding additional ways to help teams deliver faster through developer experience improvements.

Although operating in a highly regulated environment, Pfizer's leaders believe in the benefits of small teams and empowering developers to be a creative organizational force. To support this goal, the DevEx team focuses primarily on two dimensions of developer experience: feedback loops and flow state. Quarterly surveys capture KPIs such as overall satisfaction and engagement, as well as more specific factors such as codebase quality and team autonomy. The DevEx team not only analyzes aggregated results to plan its own initiatives, but also provides customized insights to individual teams.

Pfizer empowers individual teams to make their own local improvements by providing dedicated time and resources. Offering teams the opportunity to tackle improvements within their own spheres of influence allows Pfizer to transform at a much faster rate than top-down efforts alone would achieve. This innovative approach frees up the DevEx team to manage enterprisewide initiatives such as creating a unified developer portal, reducing cross-team code duplication, and improving tools for source control management.

GETTING STARTED

These frameworks provide leaders with a clear view of what to measure and focus on to improve developer productivity. Taken together, the three dimensions of DevEx—feedback loops, cognitive load, and flow state—provide a practical framework for understanding developer experience, while the accompanying measurement approaches systematically guide improvement.

Organizations should begin measuring developer experience now, even if they have not yet established or planned a formal DevEx investment. Measurement can help growing organizations spot and understand trends, decide the right time to begin making investments, and navigate macro shifts such as remote work and AI-powered programming.

For smaller organizations and startups, DevEx is critical to innovating quickly and finding market fit. As organizations evolve, DevEx fuels business performance through improvements in engineering efficiency, product quality, and employee retention. While a comprehensive view of developer productivity remains difficult to capture, measuring and focusing on developer experience provides a proven path toward building high-performing organizations.

References

1. Csikszentmihalyi, M. 2008. *Flow: The Psychology of Optimal Experience*. Harper Perennial Modern Classics.
2. Feldman, D. H., Csikszentmihalyi, M., Gardner, H. 1994. *Changing the World: A Framework for the Study of Creativity*. Praeger Publishers/Greenwood Publishing Group.
3. Forsgren, N., Humble, J., Kim, G. 2018. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations.* IT Revolution Press.
4. Forsgren, N., Kersten, M. 2018. DevOps Metrics. *Communications of the ACM* 61(4), 44-48; https://dl.acm.org/doi/10.1145/3159169.
5. Forsgren, N., Storey, M., Maddila, C., Zimmermann,

T., Houck, B., Butler, J. 2021. The SPACE of developer productivity: There's more to it than you think. *Communications of the ACM* 19(1), 20-48; https://dl.acm.org/doi/10.1145/3454122.3454124.

6. Fowler, F. 1995. *Improving Survey Questions: Design and Evaluation*. Sage Publications.
7. Gartner, Inc. 2022. Technology adoption roadmaps for IT leaders survey.
8. Graziotin, D., Fagerholm, F., Wang, X., Abrahamsson, P. 2018. What happens when software developers are (un)happy. *Journal of Systems and Software* 140, 32–47; https://www.sciencedirect.com/science/article/pii/S0164121218300323?via%3Dihub.
9. Greiler, M., Storey, M., Noda, A. 2022. An actionable framework for understanding and improving developer experience. *IEEE Transactions on Software Engineering*; https://ieeexplore.ieee.org/document/9785882.
10. Janssens, S., Zaytsev, V. 2022. Go with the flow: software engineers and distractions. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 934–38; https://dl.acm.org/doi/10.1145/3550356.3559101.
11. Jaspan, C., et al. 2020. Enabling the study of software development behavior with cross-tool logs. *IEEE Software* 37(6), 44–51; https://ieeexplore.ieee.org/document/9159122.
12. Nakamura, J., Csikszentmihalyi, M. 2009. Flow theory and research. In *The Oxford Handbook of Positive Psychology*, ed. S. J. Lopez and C. R. Snider, 194–206; https://academic.oup.com/edited-volume/28153/

chapter-abstract/212941827?redirectedFrom=fulltext.

13. Srivastava, S., Trehan, K., Wagle, D., Wang, J. 2020. Developer velocity: how software excellence fuels business performance. McKinsey & Company; https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/developer-velocity-how-software-excellence-fuels-business-performance.
14. Sweller, J. 1988. Cognitive load during problem solving: effects on learning. *Cognitive Science* 12(2), 257–85; https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1202_4.

Abi Noda *is the founder and CEO at DX, where he leads the company's strategic direction and R&D efforts. His work focuses on developing measurement methods to help organizations improve developer experience and productivity. Before joiing DX, Noda held engineering leadership roles at various companies and founded Pull Panda, which was acquired by GitHub in 2019. For more information, visit his website at abinoda.com.*

Margaret-Anne Storey *is a professor of computer science at the University of Victoria and holds a Canada Research Chair in human and social aspects of software engineering. Her research focuses on improving processes, tools, communication, and collaboration in software engineering. She serves as chief scientist at DX and consults with Microsoft to improve developer productivity.*

Nicole Forsgren *is a partner at Microsoft Research, where she leads the Developer Velocity Lab. She is lead author of* *the Shingo Publication Award-winning book* Accelerate: The Science of Lean Software and DevOps. *Her work on technical practices and development has been published in industry and academic journals and is used to guide organizational transformations around the world. For more information, visit her website at nicolefv.com.*

Michaela Greiler *is the head of research at DX, where her research focuses on developer productivity and experience. She was previously a researcher at the University of Zurich and Microsoft Research, where she focused on using engineering data to help improve developer productivity. She regularly consults for companies about how to improve their engineering teams' development practices.*

Copyright © 2023 held by owner/author. Publication rights licensed to ACM.