# DevEx in Action

## A STUDY OF ITS TANGIBLE IMPACTS

NICOLE FORSGREN, MICROSOFT RESEARCH

EIRINI KALLIAMVAKOU, GITHUB NEXT

ABI NODA, DX

MICHAELA GREILER, DX

BRIAN HOUCK, MICROSOFT

MARGARET-ANNE STOREY, UNIVERSITY OF VICTORIA AND DX

Somewhere, right now, a software developer is pulling open a ticket from the project backlog, excited by the prospect of working on something new. As the developer begins reading through the description of the task, their laptop is suddenly flooded with alerts from the team's production error-tracking system, disrupting the developer's ability to focus. Eventually, returning to the task at hand, the developer studies the requirements described in the ticket. Unfortunately, the task lacks context and clarity, so the developer asks for help, which will take days to resolve. Meanwhile, the developer checks on a previous task, which has been stuck in the queue waiting for approval for several days. The tests and builds repeatedly flake out, halting the progress of reviewers

each time they attempt to verify the changes. As the developer hops from task to task, hoping finally to immerse in some deep work, they realize that today's *experience* isn't as good as it should be to allow for their best work.

For many professional software developers, this anecdote is all too similar to their daily experiences. Friction is abundant, the development lifecycle is riddled with red tape, and successful delivery of code to production is a frustratingly infrequent event. Even worse, the problems keep compounding. Developers look on helplessly as upper management fails to intervene, leading to standstill velocity and the departure of top engineers.

How is it that organizations end up in this predicament?

Today, DevEx (developer experience) is garnering increased attention at many software organizations as leaders seek to optimize software delivery amid the backdrop of fiscal tightening and transformational technologies such as AI. Intuitively, there is acceptance among technical leaders that good developer experience enables more effective software delivery and developer happiness. Yet, at many organizations, proposed initiatives and investments to improve DevEx struggle to get buy-in as business stakeholders question the value proposition of improvements. "What is developer experience?", many of them challenge. "And why does it matter?"

## WHY DEVEX MATTERS

DevEx encompasses how developers feel about, think about, and value their work.[11] Why does this matter? First of all, developers build software and use engineering systems every day, so they are perfectly positioned to give

critical insights into how well systems and processes work. For example, is it easy and intuitive to write code and ship it to customers? Or is it confusing and full of manual steps, which are prone to mistakes and outages? Sure, you could argue that in both cases, developers are writing code and shipping software, but the environment and circumstances are very different, and they can be a leading indicator of the quality, reliability, maintainability, and even security of the systems.

DevEx is also important because of its impacts on development.

That may seem obvious because many organizations around the world, from startups to not-for-profit companies to large enterprises, employ developers to write software for customers, improve internal tools, or automate complex processes. But there is a difference between simply writing code and *writing code in an environment that is optimized for writing code*. Environments that are optimized for writing code are efficient, effective, and conducive to well-being, and rely on the right mix of tools, practices, processes, and social structures. These environments help developers:

- Get into the flow and minimize interruptions so they can focus and solve complex tasks.
- Foster connections and collaborations so they and their teams can be creative when it matters most.
- Receive high-quality feedback so they can make progress.

Considering DevEx in this light shows that development is about so much more than just writing code. It is a socio-technical process that aids developers' work while contributing to broader team performance and

organizational missions and cultures. We are not aware of empirical investigations into the impacts of DevEx; there is a need to study outcomes of developer work and the work design that supports it.

Thus, the goal of the research described here is to answer this question: How does DevEx impact individual developers, as well as their teams and organizations? Spoiler alert: An improved developer experience has positive outcomes – and not just on developers; it also helps improve team and organization outcomes. For example, we find that a better developer experience can improve productivity, learning, innovation, and profitability—and more. Read on for details, or head to the Analysis and Discussion if you just want to know now.

## DEVEX AS WORK DESIGN

Our research is based on WDT (work design theory)[22] for two reasons. First, the theory considers the outcomes of work among many dimensions. Research in WDT has found that there are important outcomes and implications for individual contributors, teams, organizations, and society. Our previous research[7] also found that by improving the work environment and work design of developers, performance outcomes are better for individuals (e.g., reducing burnout), teams (e.g., improving software delivery), and organizations (e.g., improving customer and organizational metrics).

Second, our investigation is grounded in WDT because its conceptualization of work is complex enough to account for the work practices of software developers today. WDT views work as both an assigned job—that is, a group

of tasks formally assigned to an individual—as well as "emergent, social, and sometimes self-initiated activities."[23] Software developers' work includes assigned tasks (such as items assigned during a sprint), as well as activities that emerge (such as reactive work to fix bugs, self-initiated creative work, and social activities to collaborate and improve processes).

This study uses WDT to do two things: First, it expands on our previous work to investigate broader outcomes at the individual, team, and organizational levels. Second, it explores the work design of developers—with a focus on DevEx—that positively influences those outcomes.

## Outcomes

When considering the outcomes of development work or the developer experience, many researchers and people think about productivity.[8,21] In our years of experience, however, we have seen that the improvements in developers' work go far beyond personal productivity for individual contributors,[16] to include team and organizational outcomes.[7,11] This investigation considers outcomes at the developer, team, and organizational levels, which is supported by WDT.[23]

### *Developer outcomes*

Developer outcomes are those that benefit an individual developer. In particular, prior WDT research shows that improved work design positively influences job performance, creativity,[22] and learning[5]—three outcomes investigated in this study.

### *Team outcomes*

Team outcomes are those that can benefit an individual developer but more likely accrue at the team level of work and are therefore operationalized and studied at this level. WDT also shows that outcomes such as quality benefit teams.[22] In the context of DevEx, we want to capture how work design can impact the quality of the system the team can work in, and therefore capture this as code quality and technical debt.

### *Organization outcomes*

Organization outcomes benefit a worker's employer. While developer and team outcomes likely accrue to the organization, investigating the impacts of work improvements specific to the organization is still important. This is because it can demonstrate the relationship of DevEx to the organization's mission, explain the value of DevEx to the organization, and provide evidence that can help justify and advocate for investments in DevEx initiatives. Indeed, prior WDT research has shown that improvements in work design impact organizations. Many of these outcomes are top of mind for business leaders, including retention and innovation.[22] Prior research has also shown that improvements in developer work positively affect an organization's profitability and its ability to achieve goals.[7] These measures of organization outcomes—retention, innovation, profitability, and ability to meet goals—are measured in this study.

## Developer experience

Based on our prior work, here we present a model for understanding and measuring DevEx through three dimensions that have been found to impact developer experience: flow state, feedback loops, and cognitive load.[21] This section defines each of these dimensions and describes how they are supported by WDT. Hypotheses are shown as *Hn*, meaning Hypothesis 1, 2, or 3, followed by the hypothesis we are testing.

### *Flow state*

Flow state, often described as "being in the zone," is a mental state where a person is fully immersed in work and has feelings of energized focus, full involvement, and enjoyment.[4] Achieving and supporting flow state occurs through environmental settings (e.g., quiet rooms), tooling (e.g., focus mode in tools), and personal or team practices (e.g., designating blocks of time to do deep work). Similarly, prior research in WDT found that novel work, as well as aspects of the work environment that support focus (such as discretion in scheduling and work areas that are free from noise) influence work-related outcomes.[5] We measure flow state in the following ways: satisfaction with the amount of time engaged in deep work, frequency of interruptions, and tasks that hold the developer's interest.

Based on prior developer research and WDT, we posit that flow state will have positive outcomes in the developer context and that these outcomes will come at three levels: developer, team, and organization. Stated formally:

*H1 – Flow state positively impacts (a) developer, (b) team, and (c) organization outcomes.*

For clarity, we expand this first hypothesis statement:

***Hypothesis 1a:***

*Flow state positively impacts developer outcomes.*

***Hypothesis 1b:***

*Flow state positively impacts team outcomes.*

***Hypothesis 1c:***

*Flow state positively impacts organization outcomes.*

The remaining hypotheses use the same notation and have a similar structure.

### *Feedback loops*

A feedback loop occurs when part of the system is used as input to another part of the system.[6] In the context of work and software development, the speed and quality of the information in the feedback loop are also important.[21] Prior research in WDT has found that accurate and timely feedback supports outcomes such as personal performance and catching errors.[5,1] In this study, we measure feedback loops as the time to get code changes approved and the frequency of getting a question answered quickly.

We therefore hypothesize that feedback loops support outcomes at three levels: developer, team, and organization:

*H2 - Feedback loops positively impact (a) developer, (b) team, and (c) organization outcomes.*

### *Cognitive load*

Cognitive load is the amount of information that working

memory can process at one time, and it helps with problem solving and learning.[25] In the context of DevEx, cognitive load is the amount of mental processing required for a developer to complete a task.[21] Cognitive load theory describes a framework with three types of cognitive load: *intrinsic* is the inherent amount of effort or difficulty required to do a task; *extraneous* is the way that information is presented, which can be modified to be more or less intuitive; and *germane* is related to schemas.[25]

Prior research in WDT has tested environmental and work characteristics that are well-aligned with cognitive load and that contribute to work outcomes. For example, one previous study found that easy-to-understand tasks (intrinsic) and well-designed information flows (germane) contribute to outcomes.[5] Another investigation, which was a large meta-analytic study, found that job complexity (intrinsic) and factors supporting information processing (germane) contributed to outcomes.[15]

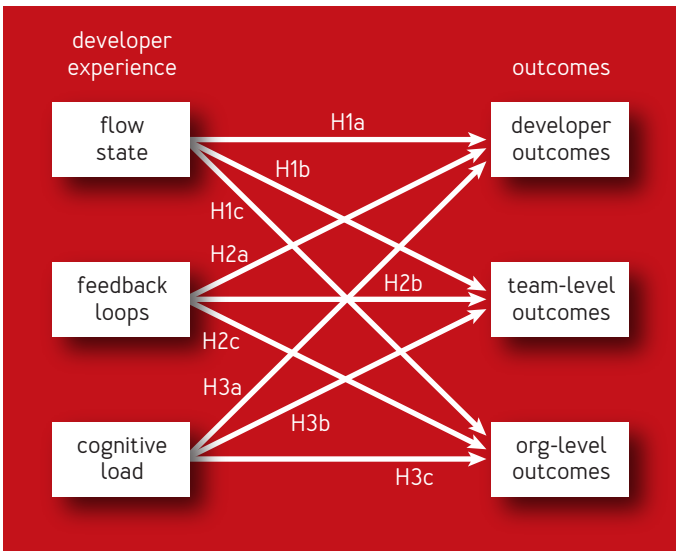
Our research measures cognitive load as the ease of deploying changes, how easy it is to understand code, and how intuitive it is to work with processes and developer tools.

Stated formally, we hypothesize that a low cognitive load supports better outcomes for developers, development teams, and their organization:

*H3 - Low cognitive load positively impacts (a) developer, (b) team, and (c) organization outcomes*

The proposed research model is presented in figure 1. The survey used to measure the model is described in the next section.

FIGURE 1: **PROPOSED MODEL**




MEASUREMENTS AND DATA

A cross-sectional survey was created using items that were previously validated in the literature, or were developed and refined over time with expert input. The items that were refined over time were developed with subject matter experts and refined over three years; this included pilot data collection, subsequent item iteration and analysis, and periodic refinements following feedback from experts and statistical analysis. Sources for all survey items are noted in table 1. This table lists items for each construct. Details for items include mean and SD (standard deviation, in parentheses). Details for constructs include CR (composite reliability) and AVE (average variance extracted). Response options and sources are

TABLE 1: **SURVEY ITEMS AND DESCRIPTIVE STATISTICS**

| CONSTRUCT | ITEM | MEAN (SD) | LOADING | CR | AVE |
|---|---|---|---|---|---|
| Flow State | I have a significant amount of time for deep work in my work days.[1, 8] | 3.383 (0.845) | 0.826 | 0.776 | 0.542 |
| | In a typical week, how often are you interrupted to work on something else that was unplanned or suddenly requested?[2, 8] | 3.826 (1.087) | 0.557 | | |
| | Generally speaking, the coding tasks I work on are more engaging than boring[1, 10] | 3.580 (0.871) | 0.796 | | |
| Feedback loops | How often does it take more than 10 minutes to obtain an answer to an internal technical question (i.e., about code, a system, or the domain you are working in)?[3, 8] | 2.799 (1.309) | 0.793 | 0.715 | 0.558 |
| | Approximately what percentage of the code reviews you request are completed within 4 business hours?[4, 8] | 2.895 (1.412) | 0.698 | | |
| Cognitive load | For the primary team you work on, how would you rate the ease of deploying changes?[5, 8] | 3.735 (0.858) | 0.728 | 0.820 | 0.534 |
| | How often can you easily understand the code you work with?[1, 8] | 3.827 (0.788) | 0.648 | | |
| | In general, the processes I need to follow to do my work are easy for me to figure out.[1, 11] | 3.607 (0.841) | 0.759 | | |
| | In general, the developer tools I have are intuitive and easy to use[1, 11] | 3.689 (0.854) | 0.780 | | |
| Developer impacts | I learned new skills related to my work in the past month[6, 9] | 3.922 (0.995) | 0.670 | 0.825 | 0.614 |
| | I have felt very productive over the past month[6, 12] | 3.680 (0.990) | 0.816 | | |
| | I have been creative in my work in the past month[6, 9] | 3.635 (0.993) | 0.852 | | |

*continues*

| CONSTRUCT | ITEM | MEAN (SD) | LOADING | CR | AVE |
|---|---|---|---|---|---|
| Team impacts | How would you rate the quality of the code you work on?[5, 8] | 3.584 (0.865) | 0.945 | 0.790 | 0.660 |
| | How often does technical debt impact your ability to complete new work?[1, 8] *(reverse coded)* | 2.826 (0.917) | 0.653 | | |
| Organization impacts | How often do you look for jobs at other companies? (Again, this question is private and only visible to the research team.)[7, 9] | 4.142 (1.024) | 0.607 | 0.823 | 0.545 |
| | My company culture supports innovation[6, 13] | 3.795 (0.999) | 0.869 | | |
| | My organization achieves its goals[6, 14] | 3.890 (0.828) | 0.830 | | |
| | My organization is profitable[6, 15] | 3.763 (0.913) | 0.605 | | |

1 1=Never, 2=Rarely, 3=Sometimes, 4=Very Often, 5=Always Have

2 1=At least once every couple of hours, 2=At least once per day, 3=At least once every two days, 4= At least once per week, 5=Less than once per week

3 1=At least once every two days, 2=At least once per week, 3=At least once every two weeks, 4=At least once per month, 5=Less than once per month

4 1=0-20%, 2=21-40%, 3=41-60%, 4=61-80%, 5=81-100%

5 1=Very Bad, 2=Bad, 3=Acceptable, 4=Good, 5=Very Good

6 1=Strongly Disagree, 2=Disagree, 3=Undecided, 4=Agree, 5=Strongly Agree

7 1=Every day, 2=Every week, 3=Every month, 4=Every few months, 5=I never look for jobs at other companies

8 Refined and adapted over three years; contact third author for details.

9 Based on experience and internal surveys; contact first author for details.

10 Adapted from Magyaródi et al. (2013)

11 Adapted from Morrison et al. (2014)

12 Adapted from Murphy-Hill et al. (2019)

13 Adapted from Meijer (2019)

14 Adapted from Theriou et al. (2017)

15 Adapted from Theriou et al. (2017)

included at the bottom of the table.

Data collection was done via a web-based survey; this was administered by DX, a company that offers a developer experience platform. The participants were developers at companies that were DX customers. Among these customers, developers are regularly surveyed regarding their DevEx (referred to in the following as the *regular survey*). Immediately upon completing the regular survey, developers were invited to participate in our study (the *research survey*).

Completing both of these surveys was optional, although only developers who had completed the regular survey were then invited to complete the research survey. No questions were duplicated between the two surveys. Across DX's portfolio, the completion rate for the regular survey is greater than 90 percent, and takes a median time of 10 minutes to complete.

During the five weeks of data collection, 2,213 participants were invited to take the research survey and 219 completed it, a response rate of 9.9 percent. The completion rate for participants who viewed the research survey was 87 percent. The median time to complete the research survey was 2.5 minutes. Because the research survey followed the regular survey and was voluntary, the low response rate could be a result of time constraints, which is often true of developers in enterprise settings.[20] Of those who completed the survey, 170 (77.6 percent) were from a company whose primary business is in technology, and 200 (91.3 percent) worked for a company with more than 500 employees (our cutoff for a medium or

large company). Because of privacy concerns, the research team did not collect participant demographics such as gender, age, or years of experience.

## ANALYSIS AND MODEL RESULTS

The proposed research model was tested using PLS (partial least squares) analysis, chosen for three reasons:[2,3] (1) It is well suited for exploratory analysis and theory building; (2) PLS does not require assumptions of multivariate normality; and (3) PLS works well with small-to-medium sample sizes. When considering CBSEM (covariance-based structural equation modeling) vs. PLS, CBSEM is more concerned with model fit while PLS is particularly well suited for predictive models,[3] which is an ideal application when considering real-world outcomes. In addition, our proposed model contains nine independent variables and nine dependent variables, with two control variables (explained later in this section), making it a fairly complex model; in comparison, CBSEM may show poor model fit simply because of the complexity of the model.[3]

A rule of thumb for determining adequate sample size when conducting PLS analysis is 10 times the largest structural equation model.[9] In this study, the largest structural equations are the developer experience constructs, where each has three paths to the outcomes. Our sample size of 219 is far larger than the minimum sample size of 30.

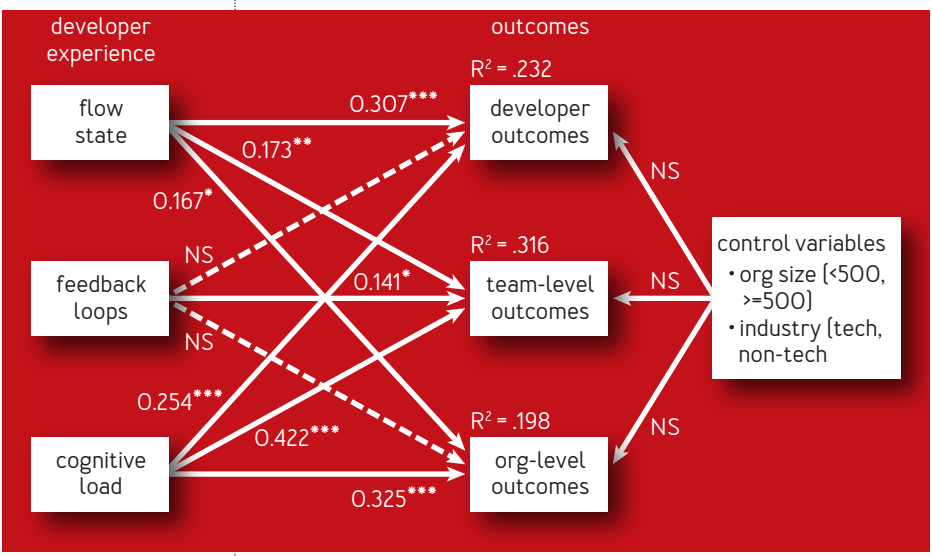
We conducted our analysis using SmartPLS 4.[24] Consistent with prior research using PLS techniques, model analysis includes two stages:[3] assessment of the (1) measurement model and (2) structural model.

In the assessment of the measurement model, convergent validity was established with three criteria:[10] (1) each item loaded on its respective construct and none below the cutoff value of 0.50 (appropriate for exploratory research);[12] (2) composite reliability of all constructs below 0.70, confirming reliability;[2] (3) AVE (average variance extracted) of all constructs greater than 0.50. Discriminant validity was confirmed by the heterotrait-monotrait ratio of correlations.[13] Thus, our measures exhibit good psychometric properties; the items and descriptive statistics are shown in table 1.

Similar to linear regression, PLS assesses the significance of relationships between constructs and provides $R^2$ values. These values indicate the proportion of variance in the dependent variable that can be explained by the independent variables. Furthermore, path coefficients and their significance can be used to assess the strength and importance of the proposed relationships between constructs. Together, the $R^2$ values and path coefficients provide insights into how well the data supports the hypothesized model.

When testing the model, we included two control variables: organization size and industry. Based on the types of companies in the sample, these were reduced to two binary values and included as dummy variables. Organization size was coded as small (fewer than 500 employees) or not-small (500 or more employees); industry was coded as primarily tech or not primarily tech. The analysis showed that the control variables were not significant. The results of the hypothesis testing are presented in figure 2 and summarized here:

FIGURE 2: **EMERGENT MODEL**




* = p < 0.05
** = p < 0.01
*** = p < 0.001
NS = not significant

- *H1* states that flow state positively impacts developer, team, and organization outcomes. This hypothesis is fully supported.
- *H2* states that feedback loops positively impact developer, team, and organization outcomes. This hypothesis is partially supported; feedback loops influence team outcomes but not developer or organization outcomes. This finding is discussed in more detail later in this article.
- *H3* states that cognitive load positively impacts developer, team, and organization outcomes. This hypothesis is fully supported.

## IPMA: Making an impact

Because of its focus on maximizing prediction of dependent variables, PLS provides additional insight into items that may have outsized impact. We conducted an IPMA (importance-performance map analysis) on the research model to identify items that could give teams and organizations additional insight. In summary, this analysis identifies items that have a high impact and performance relative to the dependent variable under consideration. To improve developer outcomes, deep work and engaging work have the biggest potential impact. To improve organizational outcomes, several items have the potential for big impact: deep work, engaging work, intuitive processes, and intuitive developer tools. We could not run the analysis for team outcomes based on the emergent model.

Note that these results are based on our research context, but they provide clues that can offer actionable insights for teams and organizations. For example, if you are hoping to improve developer outcomes such as productivity, learning, and creativity, think about ways that you can provide opportunities for deep work; these can include strategies such as encouraging both focus time for individual developers and coordinated focus time among teams, like days with few or no meetings. You can also look for opportunities to create engaging work and for learning, such as hack days.

The analysis shows that deep and engaging work provides outsized support for organizational outcomes such as innovation, retention, profitability, and broader organizational goals. Intuitive processes and tools also

support these goals. Organizations can look for ways to streamline and clarify their processes, which has been found to be impactful in other research, or provide access to intuitive, easy-to-use developer tools. Previous research has found that inefficient work processes are a top challenge for developers,[14] and improvements in process and tools present a driver for outcomes.[7]

## Alternative models

We also considered an alternative model supported by WDT. One reading of the theory allows for the items in team outcomes—technical debt and code quality—to be reframed as environmental factors that moderate developer and organization outcomes.[22] That is, they could either attenuate (reduce) or amplify (increase) the impact of DevEx factors on outcomes. A test of this model found that "team outcomes as environmental moderators"—specifically, our operationalization of technical debt and code quality—were not significant. Therefore, we do not include details of the results. Other environmental factors could be relevant. Also note that the control variables were not significant in this analysis.

## Another view of impact: Likelihood analysis

To put these results in perspective, we consider which outcomes might be expected when specific DevEx interventions are put in place. Following are the statistical results observed in this likelihood analysis, broken down by the three dimensions of DevEx: flow state, cognitive load, and feedback loops.

*Flow state*

- Developers who have a significant amount of time carved out for deep work feel 50 percent more productive compared with those lacking in dedicated time. Granted, it's not always easy for developers to reserve blocks of time on their calendars, especially if they work on teams distributed across time zones. But dedicating time to deep work is a practice that pays high dividends in terms of allowing developers to be truly productive. Encouraging developers and teams to carve out time to focus is important, and their environment needs to support this practice by minimizing interruptions.
- Developers who find their work engaging feel they are 30 percent more productive, compared with those who find their work boring. Rethinking the distribution of tasks among individuals in a team, or teams within an organization, can help here. Are the same developers continually working on less desirable projects and tasks, which could lead to burnout? Are certain teams tasked regularly with activities they find boring or divorced from the company's mission and customers?

*Cognitive load*

- Developers who report a high degree of understanding the code they work with feel 42 percent more productive than those who report low to no understanding. It's an all too familiar pattern when teams need to move fast and overlook making their code clear, simple, or well documented. While that is sometimes necessary, it can really hinder the team's long-term productivity. Tooling and conventions that help code be understandable within

and across teams can futureproof productivity.

- Developers who find their tools and work processes intuitive and easy to use feel they are 50 percent more innovative compared with those with opaque or hard-to-understand processes. Unintuitive tools and processes can be both a time sink and a source of frustration—in either case, a severe hindrance to individuals' and teams' creativity.

*Feedback loops*

- Developers who report fast code-review turnaround times feel 20 percent more innovative compared with developers who report slow turnaround times. Code reviews that are completed quickly allow developers and teams to move to their next idea quickly, setting the stage for coming up with the next great thing.
- Tight feedback loops have another positive outcome. Teams that provide fast responses to developers' questions report 50 percent less technical debt than teams whose responses are slow. It pays to document repeated developer questions and/or put tooling in place so developers can easily and quickly navigate to the response they need and integrate good coding practices and solutions as they write their code, which creates less technical debt.

## DISCUSSION

The most important contribution of this study is the evidence it provides that improving DevEx creates positive outcomes for individuals, teams, and organizations.

This is the first study we're aware of that analyzes

the statistical relationships between DevEx factors and outcomes at the individual, team, and organization levels. Although prior studies have suggested these relationships, they have not been quantified. The results reported here provide concrete evidence that can empower development teams and leaders to advocate for investment in DevEx.

## How to advocate for investment in DevEx

Most developers know they need good DevEx to do their best work. Furthermore, because so many companies today are software-driven, their ability to be profitable depends on their developers' ability to be productive and creative, and write and maintain high-quality software, with low technical debt. Even a company's ability to innovate and be profitable depends on DevEx—because if it's too hard to do daily work, it's definitely too hard to innovate.

Knowing intuitively the importance of DevEx, however, is not always enough to make a compelling case to upper management. When management rightfully asks if DevEx has an impact on business, this study can provide an answer, showing that DevEx affects the performance of individual developers, teams, and organizations. Further, our analysis clearly indicates which factors should be given priority for teams to achieve positive outcomes. This evidence can justify a DevEx initiative, as well as provide actionable insights to guide a DevEx intervention.

Now that you are sold on improving DevEx, how can you convince your organization to buy in? First, have them read this article. Then, joke aside, here are five important steps that can help you advocate for continuous improvements by keeping your arguments grounded in data.

*1. Get data on the current developer experience.* Understand what DevEx is like at your organization. For organizations that are just beginning their DevEx journey, this means collecting new data to reveal their biggest pain points, as well as knowing their current abilities to make changes. (You can use or adapt the survey used in this study, included in table 1, or a dedicated solution such as DX.) If this is your first time collecting data about DevEx, this becomes your baseline. If you have already been doing some DevEx work, you can integrate this data and update your metrics.

*2. Set goals based on your DevEx data.* Use your DevEx data to inform your goals and investments. These can be based on current business priorities, DevEx data, and what you learned from this article. For example, let's say your organization just collected DevEx data last month. It was an exploratory study, so it asked two questions: One was an NPS for internal dev tools (a 1-10 scale if the respondents would recommend the tool to others), and the other an open text question for feedback about dev tools. Using this data, you see that key challenges (opportunities!) are build times, test flakiness, and gaps in monitoring. When reviewing business priorities, your organization has been making ongoing investments in monitoring, build times, and improving PR processes. Reviewing this research, you see there are three categories of DevEx that are impactful, with deep work and engaging work having a large impact.

This would put you in a good position to set goals. Any of the items listed in the previous paragraph would be areas in which to start making investments. To get a bit

more strategic, you could identify overlaps. One strategic overlap is build times (identified in both your DevEx data and existing business priorities), which could align with feedback loops (supported by this study). Another possible strategic overlap is with monitoring (seen in both your DevEx data and existing business priorities), which could also support feedback loops for teams (a concept supported here).

*3. Set your team up for success.*

Once you've reviewed your data and set your goals, be sure to leverage mechanisms your company has in place to set you and your team up for success. This can mean setting a team OKR (objectives and key results)—or setting a shared OKR with another team to establish shared accountability. Communicate your goal with your team and organization. Revisit and check progress periodically.

*4. Share progress and validate investments.*

Share results with developers, as well as DevEx and business leaders, to evaluate and discuss the value of your investments. Reflect on which investments delivered impact, as well as what was surprising and what you learned. (Sharing surprises can be especially useful because it highlights the value of having data and acting on it, and the ability to course-correct quickly.) By periodically reassessing the state of DevEx and highlighting improvements, you can increase confidence that your investments are making an impact.

*5. Repeat the process.*

Go back to step 1 and collect more data. As you do this, reflect on your last experience to update and refine your data collection and interventions. (Remember that except when correcting for errors, adjustments in data collection should usually remain small to allow for comparisons over time.) In general, you should repeat this process of data collection and setting goals (or checking progress on large goals) every three to six months.

## Limitations and opportunities for research

While the results of this research are promising, there are some limitations, which is true of all research. First, this study was conducted among developers working at companies that were already engaging with a DevEx company (DX). This could indicate that these companies have a commitment to improving DevEx, which could bias the results when compared with organizations that do not have a similar commitment or culture. We also invited developers who participated in a DevEx survey, meaning we likely reached a population that cares about developer experience. We see this as a benefit, however, since these developers are likely more reflective about their experience.

Second, our measures were operationalized based on a model of DevEx that focuses on flow, feedback loops, and cognitive load. There may be other dimensions or definitions of DevEx that can warrant additional research. In addition, our operationalization of feedback loops focused on only a small subset of developers' work: answers to questions and code reviews. These are key aspects of how a developer works on a team, which may

explain why this only influenced team outcomes; we highlight the need to explore additional aspects of DevEx, such as feedback loops that come solely from people (conversations or discussions), purely from systems (automated builds or tests), and from people but mediated through systems (code review).

Third, this study was cross-sectional. DevEx is a complex process that happens over time and with processes that are mutually reinforcing. Future work should investigate longitudinal relationships between and among DevEx constructs and their outcomes. Our research strongly suggests that improving DevEx is worth the effort, and the impact of doing so can be measured. We invite you to share how improving and measuring DevEx factors impact outcomes in your organization.

## References

1. Campion, M. A., McClelland, C. L. 1991. Interdisciplinary examination of the costs and benefits of enlarged jobs: a job design quasi-experiment. *Journal of Applied Psychology* 76(2), 186-198. https://psycnet.apa.org/record/1991-25985-001.
2. Chin, W. W., Marcolin, B. L., Newsted, P. R. 2003. A partial least squares latent variable modeling approach for measuring interaction effects: results from a Monte Carlo simulation study and an electronic-mail emotion/adoption study. *Information Systems Research* 14(2), 189-217; https://pubsonline.informs.org/doi/10.1287/isre.14.2.189.16018.
3. Chin, W. W. 2010. How to write up and report PLS analyses. In *Handbook of Partial Least Squares*, eds. V.

Esposito Vinzi, W. W. Chin, J. Henseler, H. Wang, 655–690. Berlin, Heidelberg: Springer; https://link.springer.com/chapter/10.1007/978-3-540-32827-8_29.

4. Csikszentmihalyi, M. 2008. *Flow: The Psychology of Optimal Experience*. Harper Perennial Modern Classics.
5. Edwards, J. R., Scully, J. A., Brtek, M. D. 1999. The measurement of work: hierarchical representation of the Multimethod Job Design Questionnaire. *Personnel Psychology* 52(2), 305–334; https://psycnet.apa.org/record/1999-05792-002.
6. Ford, F. A. 1999. *Modeling the Environment: An Introduction to System Dynamics Models of Environmental Systems*. Island Press.
7. Forsgren, N., Smith, D., Humble, J., Frazelle, J. 2019. Accelerate State of DevOps Report; https://services.google.com/fh/files/misc/state-of-devops-2019.pdf.
8. Forsgren, N., Storey, M. A., Maddila, C., Zimmermann, T., Houck, B., Butler, J. 2021. The SPACE of developer productivity: There's more to it than you think. *acmqueue* 19(1), 20–48; https://queue.acm.org/detail.cfm?id=3454124.
9. Gefen, D., Straub, D., Boudreau, M. 2000. Structural equation modeling and regression: guidelines for research practice. *Communications of the Association for Information Systems* 4; https://aisel.aisnet.org/cais/vol4/iss1/7/.
10. Gefen, D., Straub, D. 2005. A practical guide to factorial validity using PLS-Graph: tutorial and annotated example. *Communications of the Association for Information Systems* 16(5); https://aisel.aisnet.org/cais/vol16/iss1/5/.

11. Greiler, M., Storey, M. A., Noda, A. 2022. An actionable framework for understanding and improving developer experience. *IEEE Transactions on Software Engineering* 49(4), 1411–1425; https://ieeexplore.ieee.org/document/9785882.
12 Hair Jr., J., Hult, G. T. M., Ringle, C. M., Sarstedt, M. 2021. *A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM)*. Sage Publications.
13. Henseler, J., Ringle, C.M. Sarstedt, M. 2015. A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the Academy of Marketing Science* 43, 115–135; https://link.springer.com/article/10.1007/s11747-014-0403-8.
14. Houck, B., Yelin, H., Butler, J., Forsgren, N., McMartin, A. 2023. The best of both worlds: unlocking the potential of hybrid work for software engineers. Microsoft and Vista Equity Partners whitepaper; https://www.microsoft.com/en-us/research/publication/the-best-of-both-worlds-unlocking-the-potential-of-hybrid-work-for-software-engineers/.
15. Humphrey, S. E., Nahrgang, J. D., Morgeson, F. P. 2007. Integrating motivational, social, and contextual work design features: a meta-analytic summary and theoretical extension of the work design literature. *Journal of Applied Psychology* 92(5), 1332–1356; https://psycnet.apa.org/doiLanding?doi=10.1037%2F0021-9010.92.5.1332.
16. Kalliamvakou, E., Forsgren, N., Redford, L., Stephenson, S. 2021. Octoverse Spotlight 2021: Good Day Project – Personal analytics to make your work days better.

GitHub Blog; https://github.blog/2021-05-25-octoverse-spotlight-good-day-project/.

17. Magyaródi, T., Nagy, H., Soltész, P., Mózes, T., Oláh, A. 2013. Psychometric properties of a newly established flow state questionnaire. *Journal of Happiness & Well-Being* 1(2), 89–100; https://jhwbjournal.com/uploads/files/acef39a197aeafb1b70cd2400037f869.pdf.
18. Meijer, A. 2019. Public innovation capacity: developing and testing a self-assessment survey instrument. *International Journal of Public Administration* 42(8), 617–627; https://www.tandfonline.com/doi/full/10.1080/01900692.2018.1498102.
19. Morrison, B. B., Dorn, B., Guzdial, M. 2014. Measuring cognitive load in introductory CS: adaptation of an instrument. In *Proceedings of the 10th Annual Conference on International Computing Education Research*, 131–138; https://dl.acm.org/doi/10.1145/2632320.2632348.
20. Murphy-Hill, E., Jaspan, C., Sadowski, C., Shepherd, D., Phillips, M., Winter, C., Knight, A., Smith, E., Jorde, M. 2019. What predicts software developers' productivity? *IEEE Transactions on Software Engineering* 47(3), 582–594; https://ieeexplore.ieee.org/abstract/document/8643844.
21. Noda, A., Storey, M. A., Forsgren, N., Greiler, M. 2023. DevEX: what actually drives productivity? *acmqueue* 21(2); https://queue.acm.org/detail.cfm?id=3595878.
22. Parker, S. K., Wall, T. D., Cordery, J. L. 2001. Future work design research and practice: towards an elaborated model of work design. *Journal of Occupational and Organizational Psychology* 74(4), 413–440;

https://bpspsychub.onlinelibrary.wiley.com/doi/abs/10.1348/096317901167460.

23. Parker, S. K., Morgeson, F. P., Johns, G. 2017. One hundred years of work design research: Looking back and looking forward. *Journal of Applied Psychology* 102(3), 403–420; https://psycnet.apa.org/record/2017-06118-001.
24. Ringle, C. M., Wende, S., Becker, J.-M. 2022. SmartPLS 4. Oststeinbek: SmartPLS. Retrieved from https://www.smartpls.com.
25. Sweller, J. 1988. Cognitive load during problem solving: effects on learning. *Cognitive Science* 12(2), 257–285; https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1202_4.
26. Theriou, N., Maditinos, D. I., Theriou, G. 2017. Management control systems and strategy: a resource-based perspective. Evidence from Greece. *International Journal of Business and Economic Sciences Applied Research* 10(2), 35–47; http://ijbesar.teiemt.gr/docs/volume10_issue2/management_control_systems_strategy.pdf.

Nicole Forsgren *is a partner at Microsoft Research, where she leads the Developer Experience Lab. She is an expert in DevEx, DevOps, and decision making, and is the 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.*

Eirini Kalliamvakou *is a staff researcher at GitHub Next, where she guides the team's strategic prototyping efforts, grounding them in user-centric research. Eirini is looking at AI's transformational impact on how developers create software. Previously at GitHub, Eirini led the Good Day Project and the 2021 State of the Octoverse Report, and has spoken extensively about developer productivity and happiness.*

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 joining 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.*

Michaela Greiler *is the head of research at DX, focusing on developer productivity and experience. Previously, she was at the University of Zurich and Microsoft Research to help boost developer productivity using engineering data. She also runs a training and consulting company that specializes in improving software engineering processes and practices for teams. For more information, visit her website at michaelagreiler.com.*

Brian Houck *is a principal productivity engineer at Microsoft focused on improving the well-being and productivity of Microsoft's internal developers. His work explores not only the technical factors that impact developer productivity, but also cultural, environmental, and organizational factors that impact the daily experiences of engineers. Over the past three*

*years, much of his research has centered on how the shift to remote/hybrid work has impacted developers.*

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.*