WHY USE A SURVEY

Now that we know our survey data can be trusted—that is, we have a reasonable assurance that data from our well-designed and well-tested psychometric survey constructs is telling us what we think it's telling us—why would we use a survey? And why should anyone else use a survey? Teams wanting to understand the performance of their software delivery process often begin by instrumenting their delivery process and toolchain to obtain data (we call data gathered in this way "system data" throughout this book). Indeed, several tools on the market now offer analysis on items such as lead time. Why would someone want to collect data from surveys and not just from your toolchain?

There are several reasons to use survey data. We'll briefly present some of these in this chapter.

1. Surveys allow you to collect and analyze data quickly.
2. Measuring the full stack with system data is difficult.
3. Measuring completely with system data is difficult.
4. You can trust survey data.
5. Some things can only be measured through surveys.

SURVEYS ALLOW YOU TO COLLECT AND ANALYZE DATA QUICKLY

Often, the strongest reason to use surveys is speed and ease of data collection. This is particularly true for new or one-time data collection efforts, or for data collection that spans or crosses organizational boundaries. The research that appears in this book was collected four different times.

Each time, we gathered data over a four-to six-week period, from around the world, and from thousands of survey respondents representing thousands of organizations. Imagine the difficulty (in reality, the impossibility) of getting system data from that many teams in that same time period. Just the legal clearances would be impossible, let alone the data specifications and transfer.

But let's assume we were able to collect system data from a few thousand respondents from around the world in a four-week window. The next step is data cleaning and analysis. Data analysis for the State of DevOps Reports is generally 3-4 weeks. Many of you have probably worked with system data; even more of you have probably had the distinct pleasure (more likely pain) of combining and collating Excel spreadsheets. Imagine getting rough system data (or maybe capital planning spreadsheets) from several thousand teams from around the world. Imagine the challenge to clean, organize, and then analyze this data, and be prepared to deliver results for reporting in three weeks.

In addition to the basic challenge of cleaning the data and running the analyses lies a significant challenge that can call into

question all of your work, and is probably the biggest constraint: the data itself. More specifically, the underlying *meaning* of the data itself.

You've probably seen it in your own organizations: Different teams can refer to very different (or even slightly different) measures by the same name. Two examples are "lead time" (which we define as the time from code commit to code in a deployable state) and "cycle time" (which some define as the time from code starting to be worked on by development to code in a deployable state). However, these two terms are often used interchangeably and are quite often confused, though they measure different things.

So what happens if one team calls it cycle time and the other team calls it lead time—but they both measure the same thing? Or what if they both call it lead time but are measuring two different things? And then we have collected the data and are trying to run the analysis . . . but we do not know for certain which variables are which? This poses significant measurement and analysis problems.

Carefully worded and crafted surveys that have been vetted help solve this problem. All respondents are now working from the same items, the same words, and the same definitions. It doesn't matter what they call it at their organization—it matters what they have been asked in the survey. It *does* matter what they are asked, and so the quality and clarity of the survey items become that much more important. But once the work of survey writing is done, the work of cleaning and preparing the data for analysis is faster and more straightforward.

In rigorous research, additional analyses (e.g., common

method variance checks) are run to ensure that the survey itself hasn't introduced bias into the results, and responses are checked for bias between early and late responders (see Appendix C).

MEASURING THE FULL STACK WITH SYSTEM DATA IS DIFFICULT

Even if your system is reporting out good and useful data (an assumption that we know from experience is quite often wrong and generally needs to be ascertained by trial and error), that data is rarely exhaustive. That is, can you really be sure it's measuring 100% of the system's behavior you're interested in?

Let's illustrate this with an example. One of the authors spent a portion of her career as a performance engineer at IBM, working on enterprise disk storage systems. Her team's role was to diagnose and optimize the performance of these machines, including disk read, write, cache, and RAID rebuild operations over various workload conditions. After working through several initiatives, "the box" was performing well, and the team had the metrics from all levels of the system to prove it. Occasionally, the team would still hear back from customers that the box was slow. The team always investigated—but the first report or two was dismissed by the team because they had confirmation that the performance of the box was good: all of the system logs showed it!

However, as the team started getting more reports of slow performance, more investigation was necessary. Sure, customers and the field could have incentive to lie, for example for discounts

based on broken SLAs. But the customer and field reports had a pattern—they all showed similar slowness. While this data-from-people didn't have the same degree of precision as the system logs (e.g., the minute-level precision in the reported response times vs. the millisecond precision from log files), this gave the team enough data to know where to look. It suggested patterns and showed a signal to follow in their work.

So what was it? It turned out that the box itself was performing exceptionally well. The team *had* instrumented every level of the stack and were capturing everything there was to capture . . . *in the box*. What the team hadn't captured was the interface. The way that customers were interacting with the box was introducing significant performance degradations. The team quickly spun up a small group to address and manage this new area, and soon the full system was operating at peak performance.

Without asking *people* about the performance of the system, the team would not have understood what was going on. Taking time to do periodic assessments that include the perceptions of the technologists that make and deliver your technology can uncover key insights into the bottlenecks and constraints in your system. By surveying everyone on your team, you can help avoid problems associated with having a few overly positive or overly negative responses.[1]

MEASURING COMPLETELY WITH SYSTEM DATA IS DIFFICULT

A related reason for using surveys is the inability to capture everything that is happening through system data—because your systems only know about what is happening inside the system boundaries. Conversely, people can see everything happening in and around the system and report about it. Let's illustrate with an example.

Our research has found that the use of version control is a key capability in software delivery performance. If we want to know the extent to which a team is using version control for all production artifacts, we can ask the team. They can tell us because they have the visibility to all of the work. However, if we want to measure this through the system, we have significant limitations. The system can only tell us what it sees—how many files or repositories are being checked in to version control. But this raw number isn't meaningful without context.

Ideally, we would like to know the percentage of files or repos that are in version control—but the system can't tell us that: it would require counting files checked in as well as files *not* checked in, and the system does not know how many files are not in version control. A system only has visibility to things in it—in this case, the use of version control systems is something that can't be accurately measured from log files and instrumentation.

People won't have perfect knowledge or visibility into systems either—but if you ignore the perceptions and experience of the professionals working on your systems entirely, you lose an important view into your systems.

YOU CAN TRUST SURVEY DATA

We are often asked how we can trust any data that comes from surveys—and, by extension, the findings that come from surveys. This may be illustrated by a thought exercise that we use sometimes when addressing groups of technologists and asking about their work. Ask yourself (or someone you know who works in software development and delivery) these questions:

1. **Do you trust survey data?** Without fail, this first question gets very little support; many in our audience sadly assume the worst in people and expect them to lie in surveys, or they expect survey writers and designers to try to "game" the questions to get the results they want—a topic we covered earlier.
2. **Do you trust your system or log data?** On this second question, there is often more support and nodding heads. We are comfortable with the data that comes from our systems because we feel confident that it hasn't been tampered with. So, we move on to our third question.
3. **Have you ever seen bad data come from your system?** In our experience, almost everyone has seen bad data in system files. While many assume the system data hasn't been tampered with, humans make systems (and therefore the data that comes from systems) and humans make mistakes. Or, if we do assume that bad actors can exist in our systems, it takes only one bad actor to introduce code that will make the system give us erroneous data.

Bad Actors and System Data

The cult classic *Office Space* is built around this premise: A bad actor introduces changes to financial software that deposits very small amounts of money (referred to as a "rounding error") to a personal account. This rounding error is then not reported on financial reports. This is an excellent example of bad system data.

If we are so familiar with bad data in our systems, why are we so trusting of that data and yet so skeptical of survey data? Perhaps it is because as engineers and technicians, we understand how our systems work. We believe we will be able to spot the errors in the data that come from these systems, and when we do, we will know how to fix it.

In contrast, working with survey data seems foreign, especially for those who have not been trained in survey writing and psychometric methods. But a review of the concepts presented in Part II of the book should demonstrate that there are steps that can be taken to make our survey data more reliable. These include the use of carefully identified measures, latent constructs, and statistical methods to confirm the validity and reliability of measures.

Compare our two cases: system data and survey data. In the case of system data, one or a few people can change the data reported in log files. This can be a highly motivated bad actor with root (or high system) access, or it can be a developer who made a mistake and whose error isn't caught by a review or test. Their

impact on the data quality is significant, because you probably only have one or a few data points that the business pays attention to. In this case, your raw data is bad, and you might not catch it for months or years, if at all.

In the case of survey data, a few highly motivated bad actors can lie on survey questions, and their responses may skew the results of the overall group. Their impact on the data depends on the size of the group surveyed. In the research conducted for this book, we have over 23,000 respondents whose responses are pooled together. It would take several hundred people "lying" in a coordinated, organized way to make a noticeable difference—that is, they would need to lie about every item in the latent construct to the same degree in the same direction. In this case, the use of a survey actually protects us against bad actors. There are additional steps taken to ensure good data is collected; for example, all responses are anonymous, which helps people who take the survey feel safe to respond and share honest feedback.

This is why we can trust the data in our survey—or at least have a reasonable assurance that the data is telling us what we think it is telling us: we use latent constructs and write our survey measures carefully and thoughtfully, avoiding the use of any propaganda items; we perform several statistical tests to confirm that our measures meet psychometric standards for validity and reliability; and we have a large dataset that pulls respondents from around the world, which serves as a safeguard against errors or bad actors.

SOME THINGS CAN ONLY BE MEASURED THROUGH SURVEYS

There are some things that can only be measured using surveys. When we want to ask about perceptions, feelings, and opinions, using surveys is often the only way to do this. We will again point to our previous example of organizational culture.

Often, people will want to defer to objective data to proxy for something like organizational culture. Objective data is not influenced by feelings or emotions; in contrast, subjective data captures one's perceptions or feelings about a situation. In the case of organizational culture, teams often look to objective measures because they want a faster way to collect the data (for example, from HR systems), and there is still a worry about people lying about their feelings. The challenge with using variables that exist in HR systems to proxy for "culture" is that these variables are rarely a direct mapping. For example, a commonly used metric for a "good" organizational culture is retention—or in reverse, the metric for a "bad" organizational culture is turnover.

There are several problems with this proxy because there are many factors that influence whether or not someone stays with a team or an organization. For example:

- If an employee receives an offer from another firm for a significant pay increase and leaves, their turnover may have nothing to do with the culture.
- If an employee's spouse or partner receives a job offer that requires relocation and your employee decides to follow

them, their turnover probably has nothing to do with culture.

- If an employee decides to pursue a different career or return to school, this may have nothing to do with the culture and more to do with their personal journey. In fact, one of the authors knows of a case where an employee worked at a very supportive, encouraging company and on a great team. It was that great team environment that encouraged him to follow his dreams and pursue a change in career so he could continue being challenged. In this case, the strong culture resulted in turnover, not the opposite.
- These measures can be gamed. If an employee's manager finds out they are actively looking for a job, the manager may lay the person off to make sure the employee is not counted in any turnover numbers. And in the reverse, if managers are rewarded for retaining team members, they may block transfers off of their teams, retaining people even when their team culture is bad.

Turnover can be a useful measure if we think carefully about what we're measuring.[2] But in the examples above, we see that employee turnover and retention don't tell us much about our team or organizational culture—or if they do, it's not what we may think. If we want to understand how people feel about taking risks, sharing information, and communicating across boundaries, we have to ask them. Yes, you can use other system proxies to see some of these things happening; for example, you can observe

network traffic to see which team members communicate with each other more often, and you can observe trends over time to see if team members are communicating more or less often. You can even run semantic analysis to see if the words in their emails or chats are generally positive or negative. But if you want to know how they *feel* about the work environment and how supportive it is to their work and their goals—if you want to know *why* they're behaving in the way you observe—you have to ask them. And the best way to do that in a systematic, reliable way that can be compared over time is through surveys.

And it is worth asking. Research has shown that organizational culture is predictive of technology and organizational performance, is predictive of performance outcomes, and that team dynamics and psychological safety are the most important aspects in understanding team performance (Google 2015).

---

[1] This, of course, assumes that you collect the data with an eye toward improvement-without telling everyone they must answer positively *or else*. That would be the equivalent of the joke: "Beatings will continue until morale improves." You would get the data you want-good responses-but it would be meaningless. One way to help encourage honest responses is to ensure anonymous data collection.

[2] For an interesting example of using retention as a way to determine the effectiveness of the interview process, see Kahneman 2011.