DevOps Metrics

NICOLE FORSGREN, PH.D.

MIK KERSTEN, PH.D.

**YOUR BIGGEST
MISTAKE MIGHT BE
COLLECTING THE
WRONG DATA.**

“

*Software is
eating the world.*

—Marc Andreessen

*You can't manage
what you don't
measure.*

”

—Peter Drucker

Organizations from all industries are embracing software as a way of delivering value to their customers, and we're seeing software drive innovation and competitiveness from outside of the traditional tech sector.

For example, banks are no longer known for hiding gold bars in safes: instead, companies in the financial industry are harnessing software in a race to capture market share. Using innovative apps, banks are making it possible for their customers to do most of their daily banking in a few swipes, from depositing checks to transferring money securely between bank accounts. Moreover, the banks themselves can improve their service in a number of ways, such as using predictive analytics to detect fraudulent transactions. Other industries are seeing similar changes: cars are now computers on wheels, and even the United States Post Office is in the middle of a massive DevOps transformation. Software is everywhere.

Leaders must embrace this new world or step aside.

Gartner Inc. predicts that by 2020, half of the CIOs who have not transformed their teams' capabilities will be displaced from their organizations' leadership teams. And as every good leader knows, you cannot improve what you do not measure, so measuring the software development process and DevOps transformations is more important than ever.

Delivering value to the business through software requires processes and coordination that often span multiple teams across complex systems, and involves developing and delivering software with both quality and resiliency. As practitioners and professionals, we know that software development and delivery is an increasingly difficult art and practice, and that managing and improving any process or system requires insights into that system. Therefore, measurement is paramount to creating an effective software value stream. Yet accurate measurement is no easy feat.

MEASURING DEVOPS

Collecting measurements that can provide insights across the software delivery pipeline is difficult. Data must be complete, comprehensive, and correct so that teams can correlate data to drive business decisions. For many organizations, adoption of the latest best-of-breed agile and DevOps tools has made the task even more difficult because of the proliferation of multiple systems of record-keeping within the organization.

One of the leading sources of cross-organization software delivery data is the annual State of DevOps Report [found at <https://devops-research.com/research>].

html).² This industry-wide survey provides evidence that software delivery plays an important role in high-performing technology-driven organizations. The report outlines key capabilities in technology, process, and cultural areas that contribute to software delivery performance and how this, in turn, contributes to key outcomes such as employee well-being, product quality, and organizational performance.

Bolstered by this survey-based research, organizations are starting to measure their own DevOps “readiness” or “maturity” using survey data. While this type of data can provide a useful view of the potential role that DevOps can play in teams and organizations, the danger is that organizations may blindly apply the results of surveys without understanding the limitations of this methodology.

On the flip side, some organizations criticize survey-based data wholesale and instead attempt to measure or assess their DevOps readiness or maturity using system data alone. These organizations, which are creating metrics based on the system data stored in their repositories, may not understand the limitations of this methodology, either.

By understanding these limitations, practitioners and leaders can better leverage the benefits of each methodology. This article summarizes the two separate but complementary approaches to measuring the software value stream and shares some pitfalls of conflating the two. The two approaches are defined as follows:

- **Survey data.** Using survey measures and techniques that provide a holistic and periodic view of the value stream.

Neither system data alone nor survey data alone can measure the effectiveness of a modern software delivery pipeline.

- **System data.** Using tool-based data that provides a continuous view of the value stream and is limited to what is automatically collected and correlated.

A COMPLEMENTARY APPROACH

Neither system data alone nor survey data alone can measure the effectiveness of a modern software delivery pipeline. Both are needed. A complementary approach to measurement can arm organizations with a more complete picture of their development and operations environment, address the key gaps of each approach, and provide organizations with the information they need to develop and deliver software competitively.

As an analogy, consider how a manufacturer may track the effectiveness of a complex assembly line. Instrumentation at each step provides data on rates of flow and defects within each phase and across the end-to-end system. Augmenting that with survey data of the assembly line staff can prove invaluable—for example, discovering that a newly deployed cooperative robot is putting more physical strain on employees than was promised by the robot vendor.

Capturing that information before higher defect rates, lower employee survey scores, or even lawsuits arise can prove invaluable. In this example, the survey data provides leading indicators to system data, or provides insights that system data might not disclose at all. Whereas assembly line manufacturing is extremely mature in terms of metrics and data collection, there is a severe lack of industry consensus on how to measure software delivery. This implies that this practice is still in its infancy. [Note that

Most organizations seem to have almost no visibility or reliable measurement of their software delivery practices.

this is likely related to the relative maturity of the fields themselves: the manufacturing discipline has been around for a long time, so those who study and measure it have had several decades to perfect their craft; in contrast, software engineering is a relatively young field, making its measurement study much less mature.) As such, it is critical for organizations to understand what they can and cannot measure with which approach, and what steps they need to take to gain visibility into their software delivery value streams.

Using the authors' collective decades of research and experience in collecting both survey and system data—confirmed by in-depth discussions with hundreds of experts at dozens of global organizations who make software value-stream measurement a key part of their digital transformation—this article outlines the measures necessary for understanding your ability to develop and deliver software.

START BUILDING A BASELINE NOW

There are several reasons why both system and survey data should be used to measure the value streams that define your software delivery processes. One of the most important is that most organizations seem to have almost no visibility or reliable measurement of their software delivery practices.

The earlier an organization starts measurement, the earlier a baseline is established and can be used for gauging relative improvement. For a small organization, applying system metrics as the initial baseline can be easy. For example, a 20-person startup can measure

MTTR (mean time to repair) using just an issue tracker such as Jira. A large organization, however, will need to include service desks and potentially other planning systems in order to identify that baseline and may not have implemented a tool that provides cross-system visibility. We recommend getting started with baseline collection immediately, and for many organizations that will mean collecting survey data while efforts to capture and correlate system data are under way.

In the absence of complete system measurements, comprehensive survey measures can provide a holistic view of your system relatively quickly (i.e., within several weeks). Contrast that with full visibility of your system provided by system-based metrics. Getting end-to-end system data can be a long journey as you first need to deploy a measurement solution across systems, and then make sure that cross-system integration is in place so that the data can be properly correlated. Modern value-stream metrics are making this easier, but for many organizations this has been a multiyear project.

While it is important to start as early as possible to get the benefits of system data, deploying survey data provides an almost immediate value and source of baseline information. This is valuable both for baselining current and future survey data, and for comparing survey with system data once in place. Therefore, it is best to capture a system baseline with survey measures now while continuing to build out system-based metrics.

What happens once you are fully instrumented with system-based metrics? You can continue using your survey-based metrics for both augmentation and

capturing additional data that's uniquely suited for survey methods.

There are still some measures that are important to software delivery, such as cultural measures, that survey-based measures will pick up and system-based metrics may miss. In addition, having both types of metrics provides opportunities for triangulation: if your survey measures provide data that is drastically different from the data coming from your systems, this can highlight gaps in the system.

Some might say such a gap is just an area where “people lie,” but if all of the people working closely with the system are lying, you might want to consider their experience as a true data point. If your engineers consistently report long build times and the system data reports short build times, could it be a configuration error in the API? Or could it be that the system-based measure is capturing only a portion of the data? Without consistently collecting insights from the professionals working with your systems, you will miss opportunities to see the full picture. The rest of this article outlines the pros and cons of each measurement type.

SYSTEM-BASED METRICS

System-based metrics generally refer to data that comes from the various systems of record that make up an end-to-end software delivery value stream. Important aspects of this data include:

- **Completeness.** Is the data captured from a particular system of record, such as an agile tool, complete enough to provide the kind of visibility, metrics, and reports that are the goal of the initiative? For example, if demonstrating

faster time to market is the goal, are enough historicals captured to derive the trend line of how quickly new products and features are delivered?

- **Comprehensiveness.** Is enough data captured across all systems of record? For example, to measure time to market for a customer request, you may need data from a customer/support tracking system, the roadmapping/requirements system, the agile tool, and the deployment tool chain.
- **Correctness.** Is the data sufficiently correlated to be correct? For example, if a support ticket and a defect are actually the same item but exist in two different systems, should the two systems be integrated in a way to indicate that these are the same item, or do you risk double-counting defects in this scenario?

System data advantages

- **Precision.** Only system-generated data can accurately show minute, second, and millisecond response times.
- **Continuous visibility.** System-generated data is particularly well suited for continuous/streaming data and realtime reporting. You can just point it to the data store and gather everything for targeted analysis later.
- **Granularity.** Data from systems can provide very granular data, allowing you to report on subsystems and components. This is useful for identifying trends and bottlenecks, but requires additional effort to create a higher-level picture of the full system. The more granular the data, the more work is required to paint a full picture.
- **Scalability.** Once the integration and visibility infrastructure is implemented, it can be pointed at all

systems. This means that the solution can be scaled from getting visibility on a single project to dozens or hundreds of projects with large amounts of data.

To use an analogy to illustrate: when building a house, a contractor may use concrete for the foundation; wood/nails/screws/drywall for the walls; wiring and plumbing; brick for the exterior; paint/carpet for the finish; plus any materials for the kitchen and bath. In order to track and monitor progress, you build in monitoring to track each piece of the construction and install it as the house is built. Once installed, each and every piece of this infrastructure (specific data) can continually provide reporting and metrics (continuous data) at subsecond intervals (precision). You can then combine and correlate (volume and scale) these to create a full picture of what is happening in your house.

System data challenges

- *Capturing behavior outside of the system.* This may be the most important yet most overlooked limitation in system-based data. An example is version control: your system can tell you only what is inside of it. What portion of the work being done is *not* being checked into a version control system? Common culprits include system configuration and database configuration scripts.
- *Gaining a holistic view.* Eventually, system-level data can provide a relatively full view of your system, but this requires full instrumentation, plus correlation across measures and maturity in reporting and visualization techniques so that teams can understand system state. This is a nontrivial task, especially if undertaken without

the right tooling and infrastructure in place. Additionally, the holistic view should include the human aspects of the process, such as the difficulty of deployments and software sprints, which are important for understanding the sustainability of work.

➤ *Capturing drifts in the system.* If any part of your system stack changes and your data collectors are not updated, your view of the system will be inaccurate. Note that this is not a characteristic of a first-class data reporting solution, but it is in some commercial systems and in many home-grown solutions, so it is worth mentioning as a condition to watch for.

➤ *Cultural or perceptual measures.* If you want to measure aspects of culture, these are perceptual and should be measured with surveys. Further, any measures that come from system databases (such as HR systems) are usually poor representations of the data you're trying to collect and will be lagging indicators. That is, they will be able to measure something only after it has happened (such as someone leaving a team or an organization). In contrast, survey measures can let you measure perceptions of culture in time to act on the information.

System-based metrics are useful, but they are not able to paint a complete picture of what is happening in your software delivery work. Therefore, it is strongly recommended that you augment your metrics with complementary survey measures.

SURVEY-BASED METRICS

Survey-based metrics generally refer to data about systems and people (such as culture) that comes from

Executives
tend to
overesti-
mate the
maturity
of their
organizations.

surveys. Ideally, these surveys are sent to the people who are working on the systems themselves and who are intimately familiar with the software development and delivery system—that is, the doers. It is better for teams to avoid surveying management and executives, because, as a recent study by Forrester shows, executives tend to overestimate the maturity of their organizations.³

Important aspects of this data include:

- *Cohesiveness.* Survey-based data is particularly good at providing a complete and holistic view of systems. This is because it can capture information about systems, process, and culture. Measure your system periodically and at regular intervals: every four to six months.
- *Correctness.* Survey design and measurement is a well-understood discipline and can be leveraged to provide good data and insights about systems and culture. By using carefully designed surveys with statistically valid and reliable survey questions that have been rigorously developed and tested, organizations can have confidence in their survey data.

Survey data advantages

- *Accuracy.* When collected correctly, survey data can provide accurate insights into systems, processes, and culture. For example, you can measure system capabilities by asking teams how often key tasks are done in automated or manual ways. When designed correctly, this provides a fast and accurate measurement that can be used to baseline and guide improvement efforts.
- *A holistic view of the system.* Surveys are particularly good at capturing holistic pictures of systems, because the

answers that respondents provide synthesize data related to automation, processes, and culture.

➔ *Triangulation with system data.* Survey data provides an alternate view into your system, allowing you to identify problems or errors when there are two contrasting views. Do not automatically discount your survey measures when this happens: there can often be cases where changes in configurations or system behavior alter the way that system data is collected, while survey measures remain true—and it is only the delta in these two measures that calls attention to changes in the underlying system.

➔ *Capturing behavior outside of the system.* In the discussion of system data, version control was used as an example of data that will be incomplete if it is collected only from your system. You can gain a more complete view of what is happening both within and around your system by using surveys. For example, are there situations where version control is being bypassed?

➔ *Cultural or perceptual measures related to the system.* Survey data provides insights into what it's like to *do* the work: organizational culture, job satisfaction, and burnout are important as leading indicators of work tempo sustainability and hiring/retention. Research shows that good organizational cultures drive software delivery and organizational performance,² and job satisfaction drives revenues.¹ Monitoring these proactively (through survey data) and not just reactively (through turnover metrics in HR databases) should be a priority for all technical managers and executives.

Let's return to the house analogy. When using system data, you can get detailed information from each piece

of the system that is reporting. This level of detail isn't possible (or realistic) when asking people through survey questions—but you can very quickly and easily get a holistic understanding of what your system or its components are doing. For example, you can reliably ascertain if the house is in a good state: anyone can report if the house is on fire, if a room is dirty or smoky, or if an event has caused damage. This data is much faster to gather than the time needed to instrument and then correlate and synthesize hundreds or thousands of data points. If your survey and system measures disagree, you have great cause to start debugging the system.

Survey data challenges

- **Precision.** While you can query practitioners about broad strokes, you should not rely on them for detailed or specific information. When you ask about deployment frequency, your survey options increase in log scale: people can generally tell you if they are deploying software on demand, weekly, monthly, quarterly, or yearly. Those frequencies are easy to confirm with system-based metrics (when available—though that is a nontrivial metric to get from systems, because it requires getting data from several systems along the deployment pipeline).
- **Continuity of data.** Asking people to fill out surveys at frequent intervals is exhausting, and survey fatigue is a real concern. It is better to limit the frequency of big data collections through surveys—say, every six months or so.
- **Volume.** The amount of data you collect is related to how often you collect it. Experience tells us that surveys should be kept to 20-25 minutes (or shorter) to maximize

participation and completion rates. There are notable exceptions: Amazon's famous developer survey was rolled out on an annual basis and took about an hour to complete, but the engineers were very interested and invested in the results, so they took the time to complete it.

➤ *Measures in strained environments.* If management has made it very clear that it isn't safe to be honest, or that the results will be used to punish teams, then any survey responses will be suspect. To quote the late W. Edwards Deming: "Whenever there is fear, you will get wrong figures." But, to be fair, system-based metrics are equally

suspect in unsafe and fearful environments, and possibly more so. Why? Because it only takes a single person with root access to slip a rogue metric into the system and a tired person on peer review or a CAB (change approval board) to miss it (as those of us who have seen the cult classic movie *Office Space* can attest). In contrast, it takes several or dozens or hundreds of people to skew survey results *en masse*.

CONCLUSION

Software is driving value in organizations across all industries and around the world. To help deliver value, quality, and sustainability more quickly,

Related articles

➡ Adopting DevOps Practices
in Quality Assurance
Merging the art and science
of software development
James Roche
<http://queue.acm.org/detail.cfm?id=2540984>

➡ Statistics for Engineers
Applying statistical techniques to
operations data
Heinrich Hartmann
<http://queue.acm.org/detail.cfm?id=2903468>

➡ The Responsive Enterprise:
Embracing the Hacker Way
Soon every company will be a
software company.
Erik Meijer and Vikram Kapoor
<http://queue.acm.org/detail.cfm?id=2685692>

companies are undergoing DevOps transformations. To help guide these difficult transformations, leaders need to understand the technology process.

This process can be illuminated through a good measurement program, which allows team members, leaders, and executives to understand technology and process work, plan initiatives, and track progress so the organization can demonstrate the value of investments to key stakeholders. System-based metrics and survey-based metrics each have inherent limitations, but by leveraging both types of metrics in a complementary measurement program, organizations can gain a superior view of their software delivery value chain and DevOps transformation work.

References

1. Azzarello, D., Debruyne, F., Mottura, L. 2012. The chemistry of enthusiasm. Bain and Company; <http://www.bain.com/publications/articles/the-chemistry-of-enthusiasm.aspx>.
2. DevOps Research and Assessment. 2014, 2015, 2016, and 2017 State of DevOps Reports; <https://devops-research.com/research.html>.
3. Stroud, R., Klavens, E., Oehrlich, E., Kinch, A., Lynch, D. 2017. A dangerous disconnect: executives overestimate DevOps maturity. Forrester; <https://www.forrester.com/report/A+Dangerous+Disconnect+Executives+Overestimate+DevOps+Maturity/-/E-RES136142>.

Nicole Forsgren is Co-founder, CEO and Chief Scientist at DevOps Research and Assessment [DORA]. She is best

known for her work measuring the technology process and as the lead investigator on the largest DevOps studies to date. She has been a professor, sysadmin, and performance engineer, and her work has been published in several peer-reviewed journals. Nicole earned her Ph.D. in management information systems from the University of Arizona, and she is an academic partner at Clemson University and Florida International University.

Mik Kersten *is the founder and CEO of Tasktop and drives the strategic direction of the company and a culture of customer-centric innovation. Before Tasktop, Kersten launched a series of open source projects that changed how software developers collaborate. As a research scientist at Xerox PARC, he created the first aspect-oriented development environment. He received a Ph.D. in computer science from the University of British Columbia, and his research interests focus on value stream architecture.*

Copyright © 2017 held by owner/author. Publication rights licensed to ACM.