WHITE PAPER | MAY 2015

DevOps Practitioner Series

"Metrics That Matter"—Developing and Tracking Key Indicators of High-Performance IT

Peter Waterhouse
DevOps Solutions Marketing

Table of Contents

Executive Summary

Challenge

Business leaders recognize that market growth and citizen-centric service delivery are dependent on the rapid delivery of high-quality software systems. Measuring success in a business context is critical, but because DevOps isn't a formal framework, organizations have little in the way of guidance to assess its effectiveness. This is problematic when organizations adopt measurement systems that are output-based or use metrics that incentivize behaviors counter to a DevOps culture.

Opportunity

When adopting DevOps, organizations should consider a new approach to identifying and implementing a DevOps measurement program. Since DevOps is about building a strong culture to optimize IT service delivery, this will involve consideration for actionable metrics that are open, transparent and promote collaboration. Additionally, and when applied with strategic management investments (e.g. a balanced scorecard), DevOps metrics can present a true picture of performance.

Benefits

With a comprehensive metrics program and supporting tools, organizations adopting DevOps can:

- Quickly measure and demonstrate in clear business terms the effectiveness of new DevOps processes such as parallel development and continuous delivery
- Align cross-functional teams around business value creation and continuous improvement
- Pinpoint capability gaps and initiate the remediation strategies needed to achieve targets and goals
- Eradicate existing practices that counteract a strong DevOps culture and inhibit the flow value to the business and customers

Section 1:

DevOps is Rudderless Without a Fresh Metrics Program

Without doubt DevOps and its focus on cross-functional collaboration is fast becoming the go-to strategy needed to rapidly deliver high-quality software solutions. Now, with growth, customer loyalty, competition differentiation and citizen engagement inextricably linked to digital transformation, building an IT organization that is agile and responsive is a prerequisite for success.

So in this transformational context, how can the success of a DevOps initiative be measured? This can be a difficult question to answer because DevOps by design isn't a formal framework and it provides organizations little guidance.

This presents a number of challenges to DevOps practitioners that if unresolved, often lead to suboptimal behaviors including:

Efficiency status-quo—Without guidance the IT team falls back to metrics it has traditionally used to demonstrate technical proficiency in meeting stability and resilience goals. Though not necessarily wrong, it should be remembered that because DevOps is supporting digital transformation, metrics should be realigned to demonstrate how new processes are impacting the business—by helping increase customer loyalty or rapidly responding to changing market dynamics and citizen behaviors.

Outputs over outcomes—Organizations gravitate to metrics that are commonly used in assessing the productivity of individual teams. These can include throughput metrics like cycle times or output-based metrics like number of features delivered or servers provisioned. While at first glance these seem well matched to DevOps, they can be counterproductive unless balanced with outcome-centric approaches that measure the results achieved against desired quality levels.

Low-hanging fruit—Organizations select metrics that are easily obtained, but not necessarily useful. Throughput and output-based metrics like those discussed above often fall into this category. Since DevOps success is predicated on cultural change, organizations also need to measure what's much harder to determine but extremely valuable—especially how the adoption of DevOps behaviors and concepts at an organizational level is improving business responsiveness.

Metrics as DevOps Anti-Patterns

Before embarking on a metrics refresh, organizations should consider all existing measures and their applicability in a DevOps context. Particular attention should be made to carefully reviewing those metrics and incentives that are counter to DevOps principles, especially those that are tangential to what should be the true objective of DevOps initiatives—delivering customer and business value. Table 1 below presents a list of metrics and practices that should be carefully scrutinized.

Table 1.

Anti-Pattern DevOps Metrics and Potential Effects

Product	Function	Features
Vanity Metrics	▪ Lines of code produced ▪ Function points created	Vanity metrics can be counterproductive since they reward the wrong types of behavior—especially if incentives are linked to the metric. Producing more code and features without validation can also inhibit more valuable activities such as refactoring and design/architecture simplification.
Intra-Team Metrics	▪ Agile team leaderboards ▪ Deployments/changes prevented	Beware of metrics that pit teams against each other and perhaps use vanity metrics as scoring mechanisms. Strike a balance with metrics and rewards that influence positive inter-team behaviors—such as code sharing, peer reviews and mentoring. Pay particular attention to metrics that promote an anti-DevOps culture, such as measuring operational effectiveness on the ability to stop releases and deployments.
Traditional Metrics	▪ Mean-time-between-failure (MTBF) ▪ FTEs: Servers	With a DevOps culture and the faster delivery of services some failure is to be expected. While not an acceptable condition, always realize that some failure will happen and responding to it is often much more important (and even less costly) than trying to prevent it.

Other Critical Considerations

When reviewing and developing DevOps metrics, it is also important to consider each against a general suitability checklist:

- **Obtainable**—Culture and behavioral improvements are important to measure, but metrics may be difficult to obtain or quantify. Seek out other related data points to help expose—e.g. review staff retention rates/transfers as an indicator of employee morale.
- **Reviewable**—Every metric must stand up to rigorous scrutiny in a business context. Carefully review metrics that can be easily collected, but add no tangible value—e.g. lines of code produced per developer.
- **Incorruptible**—Determine whether each metric can be influenced because of team and employee bias. Seek out linked incentives that can counteract DevOps benefits—e.g. SLA bonuses becoming change inhibitors.
- **Actionable**—Any metric must support improved decision making. Exposing A/B testing results can for example be a valuable way to quickly determine the effectiveness of new functionality.

Wherever possible metrics should also be shareable and have relevance across the software delivery pipeline to both development and operations. For example, continuously feeding back performance information during preproduction could be an essential indicator of nonfunctional design issues.

Section 2:

DevOps Success—Metrics That Matter

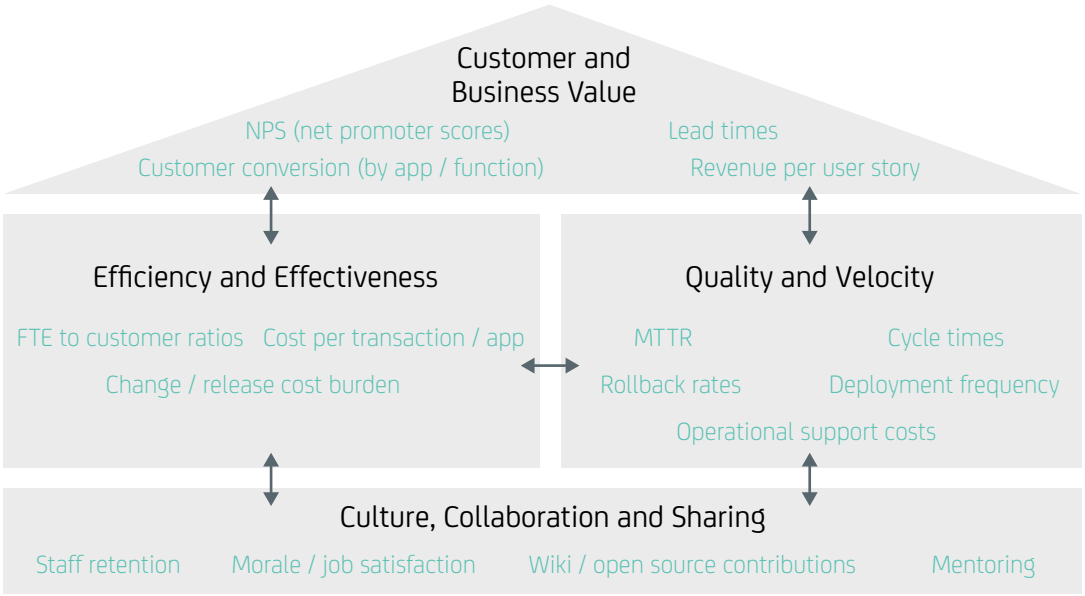
Having determined what not to measure, the next stage is to develop a candidate list of metrics supporting the DevOps initiative. One common mistake is to measure too many elements, perhaps falling back to what's easily collectable. For the most part, metrics applicable to DevOps may be new to organizations since they measure elements such as speed of deployment, rate of change, customer response and touch all aspects of people, process and technology:

- **People**—As discussed in section 1 above, people-centric metrics can be the most difficult to collect but can still be the powerful influencers on a DevOps program. Therefore, strong consideration should be given to internal metrics like staff retention rates and training, together with soft metrics such mentoring and knowledge sharing (e.g. open source contributions, wiki development).
- **Process**—Particularly relevant metrics here are those that measure the effectiveness of interlinked DevOps processes across the delivery pipeline—such as test-driven development, continuous delivery and response times. Measures here are important determine bottlenecks within the process or even identify steps within the process (or the process itself) as deficient (e.g. test readiness review gates or security audits after functional and non-functional testing).
- **Technology**—Technology-related metrics can also play an important role in DevOps, measuring such things as uptime and capacity (e.g. Can application and network capacity support the anticipated Web traffic?). Also important, are metrics that derive intelligence from failures (e.g. what is the percentage of failed releases and what percentage of these were due to code defects, manual processing, configuration errors etc.).

When developing metrics it is important to maintain a holistic view. Losing sight of this and defaulting to metrics that are skewed towards one particular aspect of measurement (e.g. operational or development efficiencies) could have a negative effect in terms of behavioral improvement. Figure A illustrates a number of dimensions which can be used to measure the overall effectiveness of a DevOps initiative.

Figure A.

DevOps Metrics Dimensions



1. Culture, Collaboration and Sharing

Metrics in this category are the cultural bedrock upon which a DevOps program is based. They are especially valuable because they provide an ongoing indicator of acceptance/resistance to new DevOps thinking. Some metrics in this dimension will be easier to collect (e.g. staff retention rates/turnover) than others (e.g. employee morale). It is important therefore to look at measures across other dimensions to understand how they impact metrics in the area—for example, are MTTR improvements impacting staff morale, absenteeism rates and responsiveness to change? Consideration may also be given to automated surveys and employee feedback—provided of course these are transparent and actionable.

2. Efficiency and Effectiveness

Metrics in this dimension normally focus on elements of development capacity and operational capabilities. While traditional metrics like server to admin ratios have been used, many organizations are now adopting more customer-centric ratios like FTE to customers. With an increased use of cloud services and automation this value should steadily increase.

Examining costs on a transactional or application basis is another good candidate metric as it's focused on improving data center efficiencies (e.g. energy, cooling). As are metrics such as cost of release, as these can expose inefficiencies associated with acquiring, preparing and maintaining physical infrastructure for development, testing and production.

3. Quality and Velocity

This dimension looks to measure data points with respect to service delivery. For organizations starting on a DevOps initiative many indicators here (e.g. **percentage of deployments rolled-back due to code defects/outages/negative user reactions**) could initially be high. This could be a result of extra time needed to adopt and make effective new processes, coupled with exposing and remediating existing technical debt and waste elements that these metrics expose. However, with DevOps' focus on establishing quality right from the start of development this should reduce over time. Metrics like this can also provide additional insights when paired with other indicators. For example, if the rate of rollbacks increases during periods of low change volume it could be indicative of more serious problems—e.g. errors due to manual/scripted release processes, task switching and excessive handoffs. Other important metrics in this dimension include:

- **Cycle time**—Not to be confused with lead time (described below), cycle time measures the length of time it takes to complete a stage or series of stages within an operation. This can be extremely useful to determine those stages (or bottlenecks) that determine the maximum speed at which the whole delivery process can run.
- **MTTR**—Measures the time taken to restore a service, which can be broken down into detection, diagnosis and recover phases. MTTR is a great indicator of how effective teams are in handling changes. For complex deployments there can be spikes, but again this metric should be trending down as DevOps becomes better established within the organization.

4. Customer and Business Value

This category of metrics are externally focused; essentially helping measure the impact of DevOps on meeting business goals, like increased customer loyalty and faster time-to-market. The manufacturing concept of lead time provides DevOps practitioners with an analogous metric (time taken from when code starts development to successful production deployment) that determines how well DevOps is meeting the need for rapid delivery of high-quality software services. This metric is especially important because long lead times are often indicative of high levels of code defects, testing constraints and contentions.

Another interesting candidate is Net Promoter Score (NPS), a simple management method to measure customer loyalty. While this metric has traditionally been used in other areas of the business (e.g. marketing) its inclusion is valid, since the loyalty of customers is increasingly determined by how quickly quality software services and updates can be delivered to customer via websites and/or mobile apps.

Section 3:

Additional Methods and Techniques

With a collection of metrics developed across each of the dimensions discussed in section 2, teams can begin a process of determining the relationships and linkages. This is important so teams can start to understand what DevOps processes and supporting tools are needed to meet targets and where capability gaps exist. Two similar methods are applicable:

1. Simple Customer and Business Impact Mapping

This methods involves determining which DevOps related processes will be needed to support a customer/business value target, together with the underpinning metrics and tools across other dimensions that will also impact that outcome. Figure B below presents a simple example:

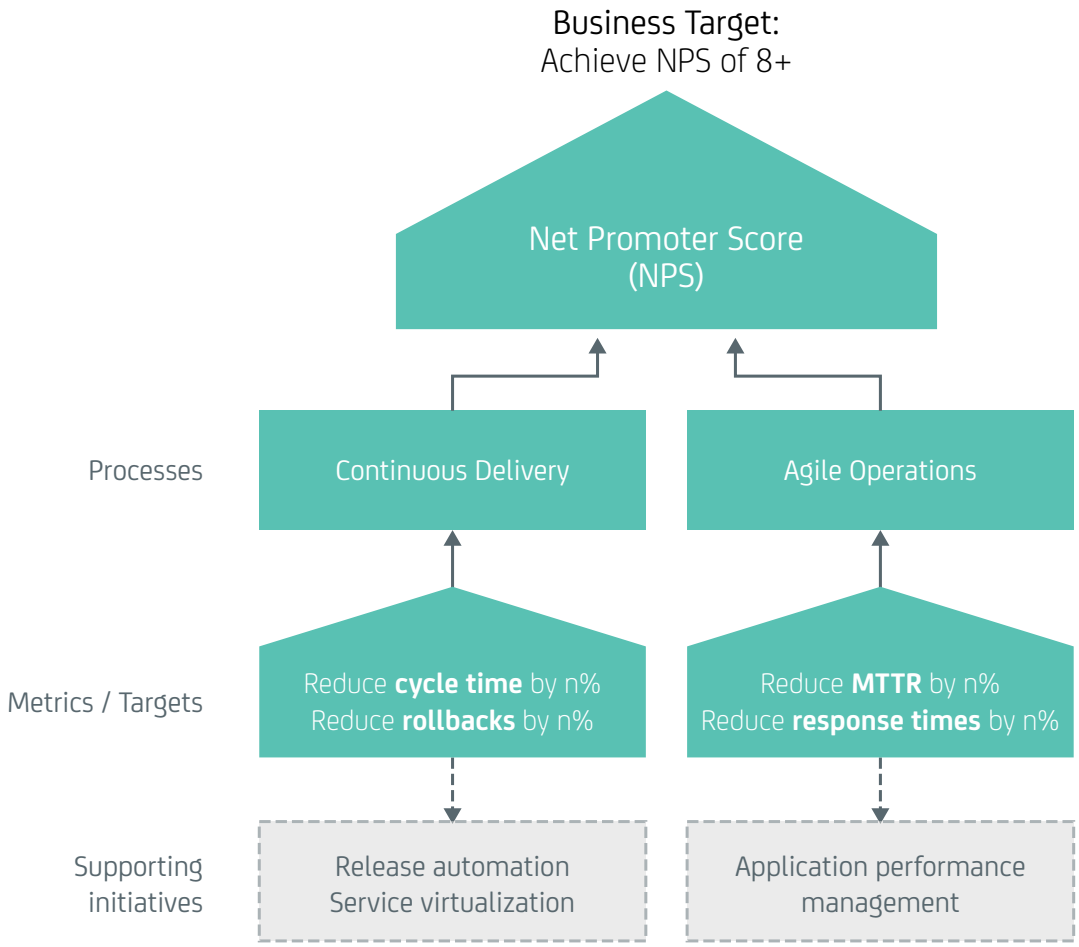


Figure B.

Metrics, Targets and Initiatives Linked to Business Outcomes

In this example that achieving a higher NPS (and realizing the associated financial benefit from increased customer loyalty) is dependent on delivering faster software releases and ensuring a top-quality customer experience. Metrics have been set (with targets) to support this outcome, with potential initiatives needed to meet and exceed goals. In the case these involve product/tool implementations but could also include organizational/team based initiatives.

2. Blended Best Practices—DevOps and Balanced Scorecard

Despite claims to the contrary, DevOps can coexist and work effectively with other best practices. Since DevOps isn't a formal framework, it can leverage existing investments, especially in areas such as performance measurement. One example is the balanced scorecard.

Originally developed by Robert Kaplan and David Norton, the balanced scorecard is a strategic planning and management system that is used by businesses, government and non-profit organizations. Its purpose is to align business activities to vision and strategy, improve communications and monitor organizational performance against strategic goals. As a performance management framework, balanced scorecard adds non-financial metrics (customer, operational and learning perspectives) to the traditional financial metrics—this provides managers with a more "balanced" view of organizational performance.

While designed as an enterprise performance tool, balanced scorecards can be useful as a framework to align a linked series of the DevOps metrics discussed in this paper to the organization's mission and business objectives—see Figure C.

Figure C.

Balanced Scorecard With Sample DevOps Metrics, Targets and Initiatives

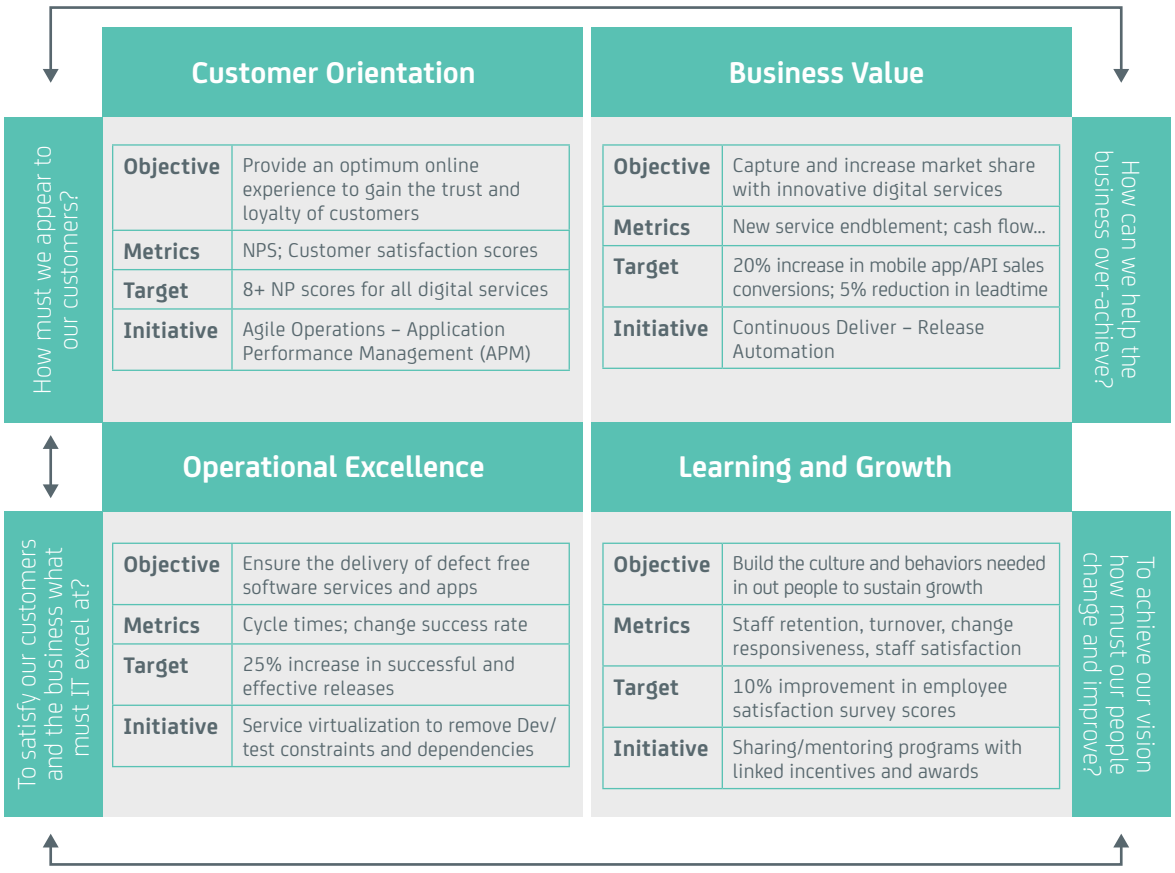


Figure C also illustrates that each metric doesn't operate in isolation. To get a true picture of performance each of the four scorecard dimensions are linked by metrics, targets and initiatives. For example, increasing change success rates by using service virtualization (removing testing constraints = faster defect free releases) will not only help improve the effectiveness of IT, but can also positively impact loyalty building objectives within the customer domain.

Continuous Improvement

When developing DevOps metrics programs practitioners should also:

- Have regular and ongoing target reviews to ensure that goals are not completely unrealistic or that existing processes and tools are not delivering improvements
- Consider removing persistent "green light" metrics when targets have been consistently achieved
- Avoid having every metric focused on velocity, defect counts and function counts without paying attention to delivery, customer satisfaction and loyalty
- Strive to prevent vanity metrics and operational or team-centric bias creeping back into the program and distorting the true performance picture
- Beware of ranking teams based on targets—the best way to compare teams is to measure things like customer loyalty (as described above) and how successful teams are in meeting their commitments
- Give strong consideration for metrics, targets and initiatives that foster peer review and openness—carefully building incentives and reward programs that help reinforce the value of a strong collaborative culture
- Involve business counterparts right from the start to ensure customer and business data is held to the same standard as operational/efficiency metrics
- Match tools to the DevOps program, especially those that can monitor and respond to real-time conditions (such as transaction times, response times, mobile app crashes) but can also proactively detect and prevent adverse conditions (such as code defects, release bottlenecks) impacting performance

Section 4:

Conclusions

Without good metrics, processes and supporting tools, organizations adopting DevOps have no way of measuring program effectiveness in context of their digital transformation goals. This can result in IT relying on metrics and tools that are output centric or skew and distort the true performance picture.

A comprehensive DevOps metrics program with the right tools will address these issues—this includes measuring efficiency and velocity (release frequency, failure rates and MTTR), but as discussed in the paper, incorporating measures, targets and initiatives to help drive organizational improvement and support business and customer-related goals.

Supporting tools should have the scope needed to present detailed and accurate information, but also quickly pinpoint and prevent potential problems across linked DevOps processes—especially those that impact an organizations ability to meet and exceed targets.

About the Author

Peter Waterhouse has been involved in the development, support and marketing of enterprise software solutions for more than 20 years. He has held a number of management, consulting, technical sales and marketing positions in areas such as cloud computing, DevOps and IT business management. Peter writes and blogs on a range of disruptive business and technology trends, with articles appearing in publications such as Information Week, Wired Insights, DevOps.com and SC Magazine.