The Four Key Metrics

Celia Melendi - UO276077

Landon Brown - UO293019

Sebastian LH - UO277069

Introduction

The four key metrics are a means of measuring software delivery performance, first identified by the DevOps Research and Assessment team (DORA). They are based on principles that successfully measure software delivery performance. Firstly, they focus on team rather than individual performance, and secondly, they prioritize outcome rather than output. By following these principles, the four key metrics are ensured to be effective measurements of software delivery performance.

There is an open-source project called Four Key that may be used to gain insight and empirical data of team performance according to these metrics. It sets up a data ingestion pipeline from a GitHub or Gitlab repository and then aggregates the data into a dashboard where the Four Key metrics are displayed and can additionally be tracked over time. The Four Key metrics are as follows:

1. **Deliver Lead Time**

Lead Time is the amount of time it takes to go from a customer making a request to the request being fulfilled. When linked to a GitHub project, it is measured by the time from the request's initial 'commit' to its deployment. However, in reality the timeline is more arbitrary. Delivery lead time consists of two phases, the first being the time it takes to develop the software, and the second being the time it takes to deliver it. When lead time is optimized (shorter is better in this case), it allows for faster feedback, course correction, and problem fixes on the delivered software.

The first phase consists of the time spent designing and developing new products and services, and as these processes are subjective, it's hard to define where this phase stops or ends. Although the lead time metric is calculated by the difference between a feature's commit to its deployment time, in actuality there is much that happens leading up to the initial commit, which is encompassed in this first phase. Since new features may require design and implementation that have never been done before, this phase may take much longer than anticipated. In turn, time estimates for this phase are uncertain, and outcomes are variable. Although DevOps techniques may be employed to improve the efficiency of this phase, the experience and skill of the engineers have a much greater impact, and even then, are not able to guarantee results.

The second phase consists of the time spent building, testing, and deploying new services. When this phase is optimized, it enables a flow between from deployment to production. This is done by utilizing techniques such as standardizing work, reducing variability, and reducing batch size. As a result, software delivery cycle times are predictable, and outcomes have low variability. This ensures stakeholders that the software team is reliable, competent, and worthy of dedicating their resources to.

Elite performers	High performers	Medium performers	Low performers
Less than one day	Between one day and a week	Between one week and a month	Between one month & six months

2. Deployment frequency

Deployment frequency is how often an organization successfully releases to production. It uses the amount of deployments per day as reference. It was based on the measure and control of the batch changes in the product. Usually, the successful the company, the smaller (amount of change) and more frequent deliveries (deployments).

It aims to reduce batch sizes and by doing so, it reduces cycle times for variability, which eventually leads to accelerated feedback, reduction of risks together with overhead and efficiency improvements. Furthermore, it also reduces costs and promotes schedule growth.

It only considers frequency of deployments because if it was batch size, it would be hard to measure in terms of software production.

In order to calculate it, teams should stablish what constitutes a successful deployment.

Elite performers	High performers	Medium performers	Low performers
On-demand (multiple deployments per day)	Between once per day and once per week	Between once per week and once per month	Between once per month and once every six months

3. Mean time to restore

The mean time to restore is how long it takes an organization to remediate a production failure. One thing must be clear, failure is inevitable, we cannot predict when is it going to happen, but we can take the necessary actions to solve the question of how quickly service can be restored, encouraging our teams to build robust systems.

In order to know the value of this metric, we need to know when the incident was issued and when it was resolved. It is measured by the average time between a bug report and its corresponding fix deployment.

Having a small service restoration time really makes the difference as it makes it more comfortable for companies to experiment and innovate their products, which leads to business revenue improvements.

Elite performers	High performers	Medium performers	Low performers
Less than an hour	Less than one day	Less than one day	Between one week/one month

4. Change fail rate

The Change Failure Rate is the percentage of deployments causing a failure in production, which means that it considers both how many deployments were attempted, and how many of those resulted in failures in production. This metric links the total count of deployments to

incidents, so the most important part of tracking this metric is identifying what constitutes a failed deployment or release. In most cases, a failure means that a deployment either results in degraded service or subsequently requires remediation.

The same practices that enable shorter lead times correlate with a reduction in change failure rates (test automation or working in small batches) since all these practices make defects much easier to identify and remediate. One of these practices is having a development process completely automated, consistent and reliable.

A key component of DevOps is the adoption of Lean practices, in particular the idea of failing fast, meaning detecting a risk of failure as early in the development lifecycle as possible as this will increase the system's quality.

Tracking and reporting on change failure rates isn't only important for identifying and fixing bugs, but to ensure that new code releases meet security requirements, as it exposes the frequency in which new code negatively impacts the system's performance.

- Elite, High, Medium performing teams: 0-15%
- Low performing teams: 46-60%

Conclusion

At a high level, Deployment Frequency and Lead Time measure velocity, while Change Failure Rate and Time to Restore Service measure stability.

And by measuring these values, and continuously iterating to improve on them, a team can achieve significantly better business outcomes, improving the efficiency and effectiveness of your own operations. The end goal, delivering value to the customer faster, is achieved through continual improvement.

DevOps metrics are data points that directly reveal the performance of a DevOps software development pipeline and help quickly identify and remove any bottlenecks in the process. These metrics can be used to track both technical capabilities and team processes.

Knowing where the team started on their DevOps Journey and comparing it to where they are now can determine if progress is moving in the right direction. Reviewing these metrics often allows for a fast feedback loop to reinforce good patterns or pivot when experiments do not bear fruit.

References:

https://itrevolution.com/measure-software-delivery-performance-four-key-metrics/

https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

https://www.atlassian.com/devops/frameworks/devops-metrics

https://www.wwt.com/article/four-key-devops-metrics-and-how-to-measure-them

https://services.google.com/fh/files/misc/state-of-devops-2019.pdf