VOLUME 1

SUZIE PRINCE
ARAVIND S.V.

ACTIONABLE CONTINUOUS DELIVERY METRICS

GoCD is an on premises, open source, continuous delivery tool with comprehensive pipeline modeling and value stream map to help you get better visibility into and control of your teams' deployments.

gocd.org

PART 1

WHY MEASURE

PART 2

WHAT TO MEASURE

PART 3

HOW TO MEASURE

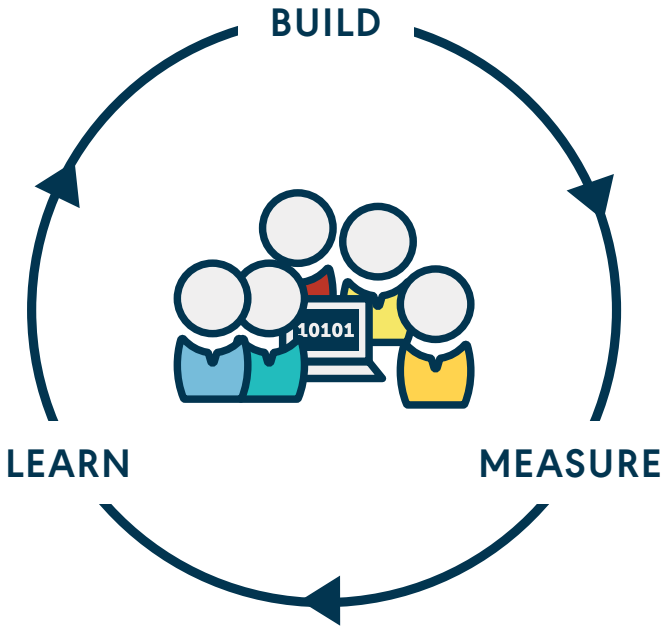
PART 4

ACT ON YOUR METRICS

Software and IT are key drivers for innovation in most organizations, and speed of software delivery is key to their success. Continuous delivery offers an accelerated feedback loop that helps deliver software faster and more reliably.

Measurement, Feedback, Improvement

The main reason why you should measure your CD process is to see if you are improving and delivering on your goals. A feedback cycle with "build, measure, learn" metrics is a way to set specific and measurable goals, direct activities towards achieving those goals, and help you understand if you are achieving those goals.



The specific data you gather can also be very useful for you to understand a number of things:

Predict Future Behavior

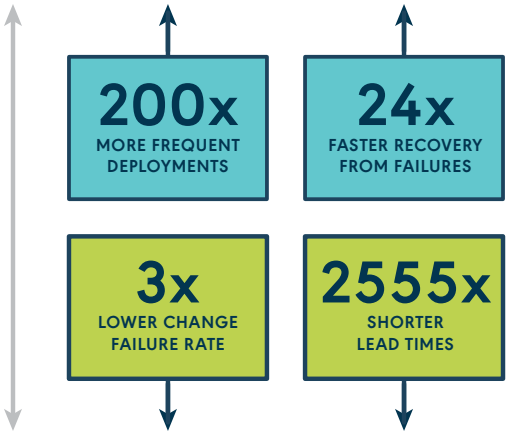
Data will provide you more accurate estimates for your business. For example, if you know your cycle time, you can more accurately answer questions about how long it will take for something to be ready for your customers.

Estimate the cost or value of an activity

If you are considering parallelizing tasks or removing manual steps in your process, once you have some data about your current process you can calculate the time savings of these improvement activities. From there, this data could potentially help your organization estimate dollars made or saved by certain specific improvements.

Continuous Delivery Baselines and Benchmarking

Once you have some data, the values can be used as your baseline. Those baseline values are key to understanding whether you are improving your own process as well as key to understanding where you stand relative to "high performing" teams.



Credit: https://puppet.com/resources/whitepaper/2016-state-of-devops-report

Once you introduce a CD pipeline and have established your path-to-production, the next step is to monitor its efficiency. There is no template for this, you'll need to decide which metrics are right for your organization.

We do not suggest measuring everything, however, here is a list that we recommend using to help monitor your CD process.

- **Throughput**
- **Cycle time**
- **Failure rate**
- **Mean time to recover (MTTR)**

There are other metrics for you to consider depending on your goals, and in this booklet we have only highlighted the ones we think are most important to get started. Some other metrics that are worth taking a look at are **defect fix times, escaped defects, total regression test time, number of branches in version control, production outages during deployment, deployment frequency, lead time, mean time between failures, and more.**

If you are selecting your own set of metrics, be thoughtful about what metrics you choose to capture and measure. Some metrics are easy to measure but provide no value to your organization.

Beware of:

Vanity metrics - Metrics that are easily manipulated and do not necessarily correlate to the numbers that really matter. These are metrics that are easy to measure but provide no value to your organization. A good example of this is bugs fixed. We see many organizations tracking bugs or defects fixed but with little understanding about whether this really provides value to their processes or their customers.

Unclear metrics - Be sure to define what your metrics mean to you (as we have done here), and educate everyone about what these mean. For instance, cycle time can have many definitions. It can mean from idea to completed feature, or commit to production. The best thing to do is stick to whichever definition you've decided on and ensure everyone on your team is clear what that is.

Invisible metrics - Make sure your metrics are visible. We suggest big visible dashboards in your workspaces and regular reports (that people read!) to ensure that the metrics really are used.

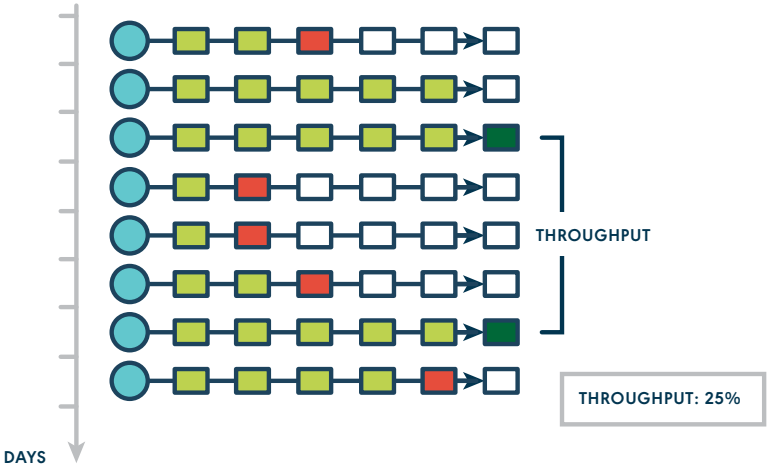
Comparing across teams - Avoid insisting every team use the same metrics if their goals differ and avoid comparing across teams and creating valueless competition.

The Hawthorne Effect or observer effect - This is when individuals modify their behavior in response to their awareness of being observed or measured. We know of an organization who started to measure test coverage and in response people began writing tests without assertions to get the test coverage results they needed.

Gathering "all the data" and not using it - Avoid gathering data for the sake of it. Be specific and clear about what is important and change what you measure if it doesn't work to help you improve.

Throughput :

Throughput is a measure of how frequently your team deploys code.



This metric is represented as a percentage.

The word "continuous" in CD implies high throughput. Having a higher throughput means that you have more deployments, increasing your confidence in your deployment capability. This also means that you're delivering value to end users and stakeholders more quickly. Generally this means your are creating more value for your organization more often. It also means you have many more opportunities for feedback on your software.

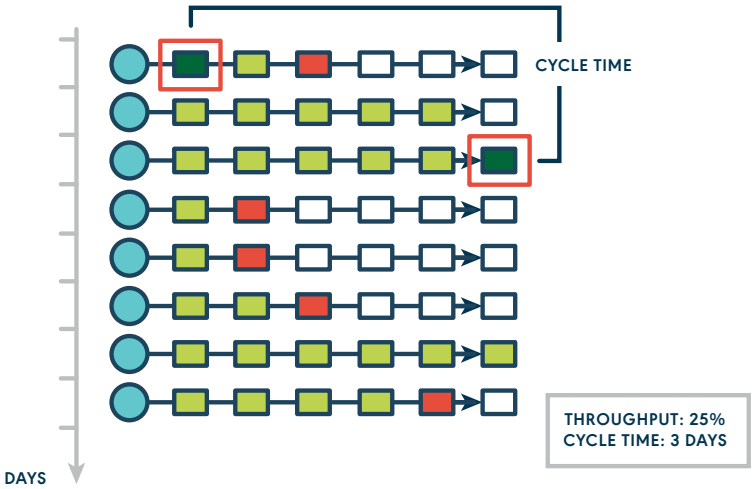
Throughput still has to be balanced with quality. You don't want to increase throughput by removing tests. You want to be able to deliver more often to production, while maintaining or even improving quality. That's what CD is about and what the throughput metric captures.

What to look for?

High functioning teams tend to have higher throughput as compared to their less efficient peers. It is good to baseline your throughput, and try to increase it as much as it makes sense in the context of your organization's business and goals.

Cycle time:
Cycle time is a measure of how long it takes from committing code to deploying it to a production environment.

From the time you commit code to that code successfully running in production, how long does that take?



This metric is represented as a duration.

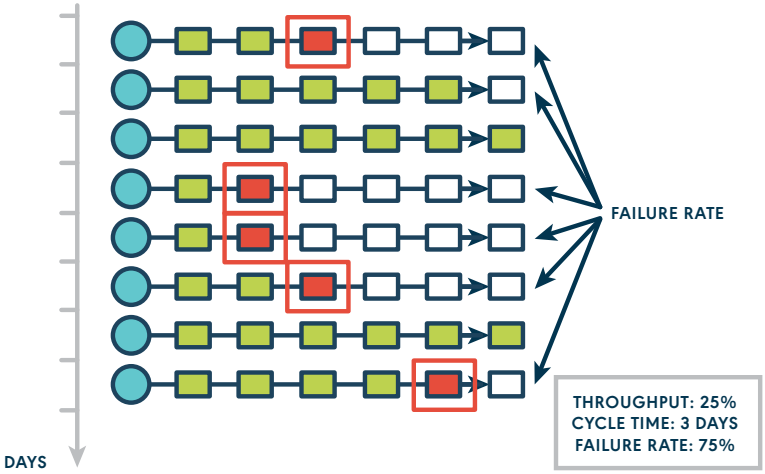
Whereas throughput is the number of times something happened, cycle time is the cumulative lapsed time it took for that to happen.

In the DevOps literature you may see *cycle time* used interchangeably with *lead time*. There is a lot of confusion around the meaning of these two terms. It is important that you are clear what you mean, you define what you mean, and you are consistent with your definition within your organization.

What to look for?
In general we aim for a short cycle time. Highly efficient teams have lower cycle times.

Failure rate:

Failure rate is a measure of the percentage of changes that result in a failure. A failure is defined as something that requires remediation such as a roll forward.



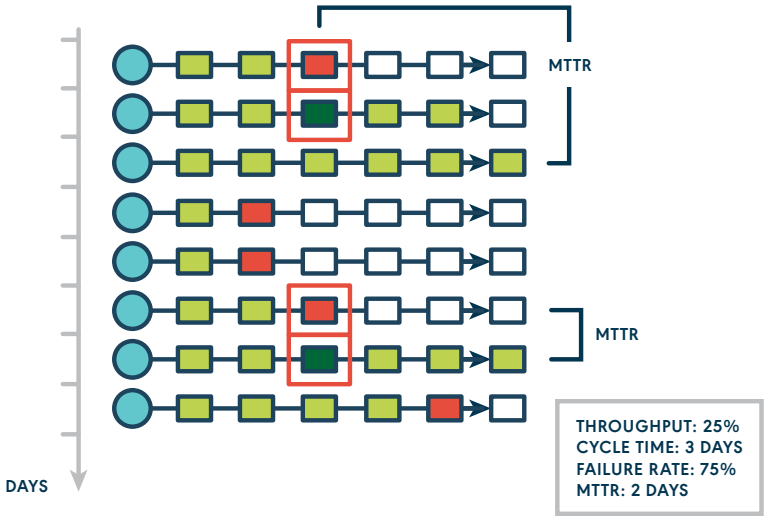
This metric is represented as a percentage.

When you increase the throughput, there's always the risk of impacting the quality of the artifact generated. This means that you're not delivering value to customers more frequently even if you deploy often. It is important to monitor both the throughput, cycle time and quality to ensure you're not introducing instability into your CD process by increasing the throughput.

What to look for?

High failure rates can be associated with longer cycle times and increased MTTR. Pay attention to whether the failures at certain stages are repeated and look to remove the underlying cause. To achieve high throughput you should reduce failure rate (but not quality) and reduce your cycle time.

Mean time to recover (MTTR):
Mean time to recover (MTTR) is a measure of how long it takes to fix a build failure. Additionally, how long it takes you to restore service during a deployment.



This metric is represented as the *mean* duration.

When looking at mean time, we look at multiple occurrences of the "failed, succeed" cycle to understand how long it took for a failure to be resolved.

What to look for?
A low MTTR implies quicker feedback. You're more open to experiments and learning quickly because recovering from failures is painless. A high MTTR means that the rest of the team has to wait a long time for you to get the pipeline or application back to a working state after any failures, elongating the feedback loop and causing them to have to wait before they make more changes/improvements. That could make you more cautious about making changes, for fear of failures.

Improvement Checklist

Here's a brief list of things you can do to get started in introducing metrics into your continuous delivery workflow.

☐ **Create a Value Stream Map -** A visual representation of your end-to-end CD workflow

☐ **Select metrics that are important to you -** Start with MTTR, Failure Rate, Throughput, and Cycle Time

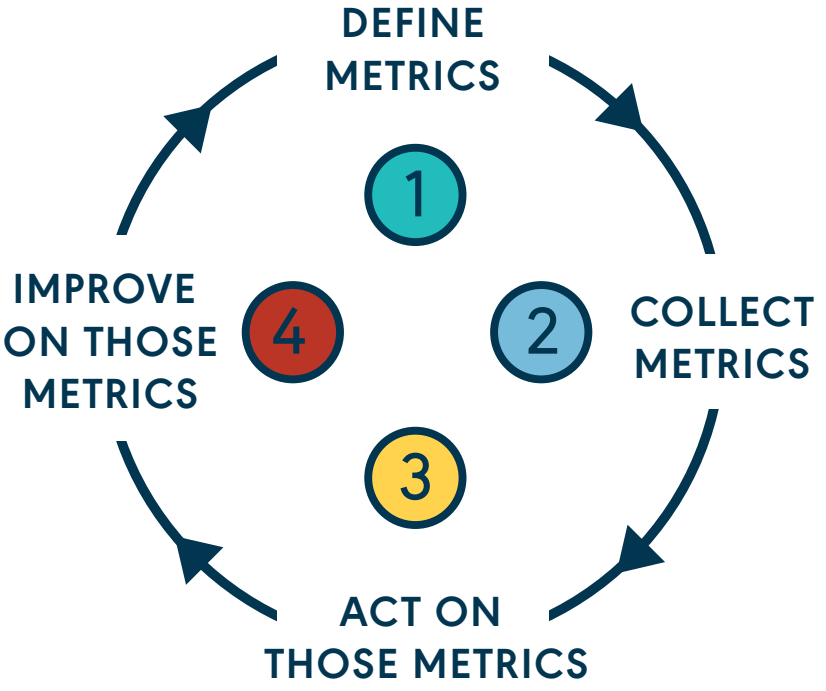
☐ **Capture your metrics -** Ensure your CD tool lets you do this, with GoCD, you can use the Analytics Plugin

☐ **Review metrics -** Once you've captured data, ensure it is visible and share it

☐ **Learn! -** Use the metrics you've collected to understand if you're achieving the goals you've set for your CD processes

Recap:

- Metrics are important to setting goals, improving and predicting
- Start with throughput, cycle time, failure rate and MTTR
- Be thoughtful about what you measure
- Look for connections between metrics
- Understand your context
- Review, change, and improve your process
- Consider using tools to help capture and visualize data



2016 State of Devops Report
https://puppet.com/resources/whitepaper/2016-state-of-devops-report

2017 State of Devops Report
https://puppet.com/resources/whitepaper/state-of-devops-report

Value Stream Map (VSM)
https://docs.gocd.org/current/navigation/value_stream_map.html

4 Important Metrics for Continuous Delivery
https://www.gocd.org/2018/01/31/continuous-delivery-metrics/

Martin Fowler Continuous Delivery bliki
https://martinfowler.com/bliki/ContinuousDelivery.html

CD of Microservices - Remediation Strategy
https://www.gocd.org/2018/09/11/cd-microservices-remediation-strategy/

ACTIONABLE METRICS TO IMPROVE YOUR SOFTWARE DELIVERY

GoCD is offering a free trial for our enterprise analytics plugin, which gives you access to the metrics covered in this booklet and allow you start improving your software delivery processes.

- How long do pipelines take?
- How quickly do we recover from failures?
- Which jobs are slowest?
- Do we need to add more resources?

Sign up at gocd.org/analytics