ConnectALL

# Ultimate Predictability Metrics Guide

www.connectall.com

+1 800 913 7457

sales@connectall.com

# What Gets Measured Gets Managed

## Move closer to predictability with tangible purpose-driven metrics

Struggling to keep your head above water with no proper visibility into software product delivery numbers? You’re not alone. In most IT organizations silos often obfuscate the big picture, and there are so many stakeholders involved that it feels impossible to have predictable software delivery.

You need a set of metrics that’ll help you measure your product delivery initiatives to make decisions about future performance and value. But, how do you measure value in software delivery?

Consider you fit into one of these profiles. And you have to work with the other two.

### Agile Teams

- Does the team have everything they need to perform the work?
- Can we meet the release commitment?
- Can the team control their WIP?
- Can the process catch issues?
- Is the code testable, malleable, and maintainable?
- Are we incurring technical debt?

**Metrics that matter:**
• Throughput
• Distribution
• Throughput variation
• Flow load
• Bottlenecks
• Escaped defects
• 80th percentile lead time expectation

### Product Teams

- Are defects being addressed in a timely manner?
- Is the team’s throughput or velocity stable?
- Are we over or under spending on maintenance?
- Are we controlling scope?
- Is the next release on track to be delivered on schedule as planned?

**Metrics that matter:**
• Throughput variation
• Distribution
• Bug aging
• Mean time to recover
• Escaped defects
• Bottlenecks
• 80th percentile lead time expectation

### DevOps Teams

- Are we meeting our uptime expectations?
- Can the team frequently deliver working, tested, remediated code?
- Will the team meet their SLAs?
- Can we release a minimum viable product?
- Are we able to recover quickly?

**Metrics that matter:**
• Throughput variation
• Flow load
• Deployment frequency
• 80th percentile lead time expectation
• Mean lead time for changes
• Mean time to recover

All three have similar problems — the solution to which is a set of metrics that are interrelated. When you diagnose one, you have a better understanding of the others. When your actions change one, it affects the other. How do you find a balance between all the metrics that ultimately will let you keep a check on the health of your value stream?

Here is a set of what we call ***The Ultimate Predictability Metrics*** that will let you have control over your value stream. These are the metrics that you must be tracking, in real time, in order to predictably deliver valuable software to your customers.

**We recommend that you look at three types of measurements: trends, specific numbers, and variation.*

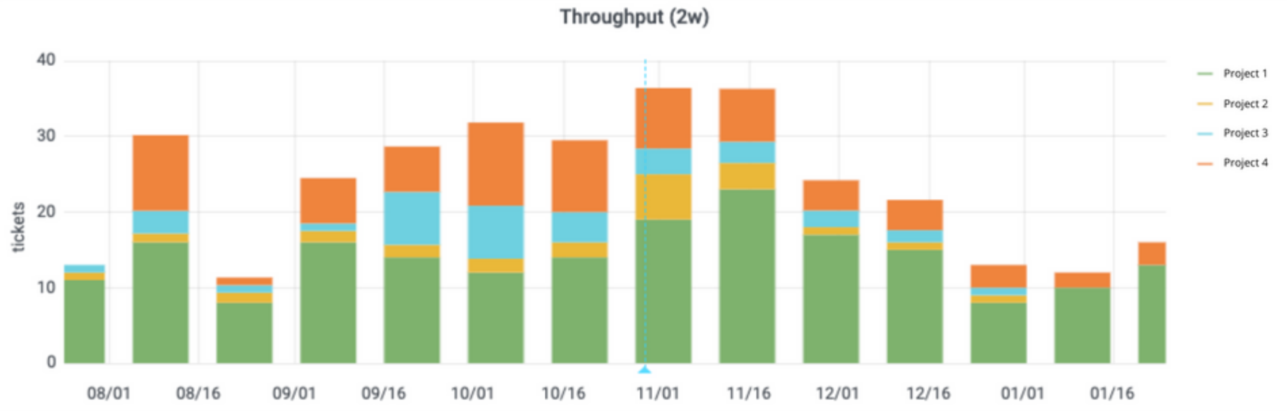
## Throughput

### What does this tell you?

- The number of distinct pieces of user functionality (tickets, bugs, stories, etc.) delivered in a specific period of time, preferably weekly or biweekly.
- A way to track performance and productivity over time.
- An unstable throughput or a variation in throughput would require process or people changes.

### Why do you need it?

- Predict the future or the done date.
- Plan out the value the team is capable of delivering.
- Check if process changes are positively or negatively affecting the rate of work.
- A stable throughput indicates the health of your value stream in terms of predictably delivering working software.



# Throughput Variation

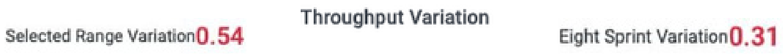
### What does this tell you?

- Variation in throughput indicates an unpredictable value stream.
- Enables you to understand how accurate the delivery date could be.*
- Variation can be computed as the standard deviation divided by the average. We recommend including data from at least the last 4 periods, but no more than 6 months of data.

**For software development and delivery teams, we recommend aiming for a closer delivery date.*

### Why do you need it?

- Stabilize the team's performance.
- Manage risks and dependencies before starting the work.
- The goal is to see a reduction in the standard deviation of your Throughput over time.
- Analyze a source of the Throughput Variation.
- Maintain a low throughput variation and a trend that is stable to increase software delivery predictability (lower than 0.20; 0.30 or above is very high).



Throughput Variation

Selected Range Variation 0.54

Eight Sprint Variation 0.31

# Escaped Defects

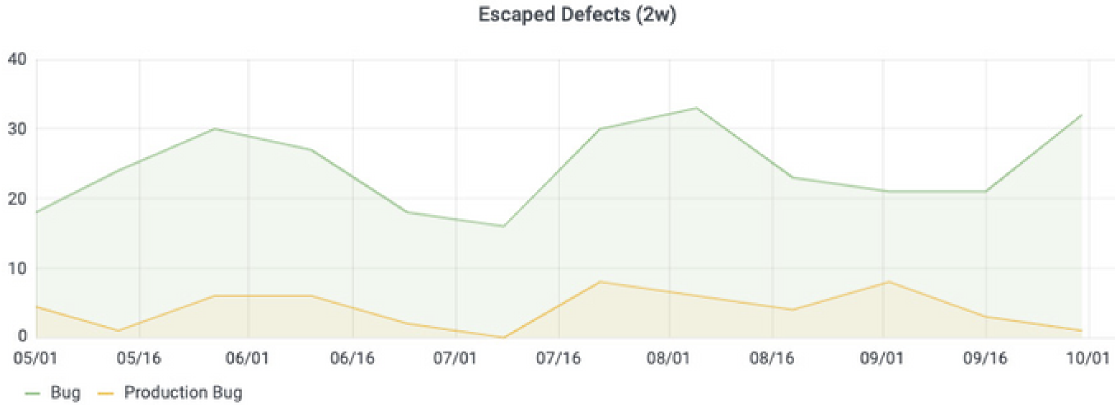
### What does this tell you?

- Tracks defects that have escaped after the end of a sprint. This means the defects were caught before production but after the end of a sprint either during integration or regression testing.
- Helps to track the defect escape rate.
- Helps understand trends and the quantity of defects introduced and how to mitigate them.

**Many organizations measure production defects, which only considers post-production defects. We recommend also measuring escaped defects to improve the effectiveness of your pre-production processes and decrease rework.*

### Why do you need it?

- Keep a check on the trends and see if your defects are decreasing.
- Determine if the changes you make to stabilize Throughput and if that is increasing pre-production defects.
- Understand how good of a job your team is doing as a whole.
- Review what type of defects are typically slipping through the cracks and which stages they are getting through.
- It is a good way to know if you are sacrificing quality for speed.



# Bottlenecks

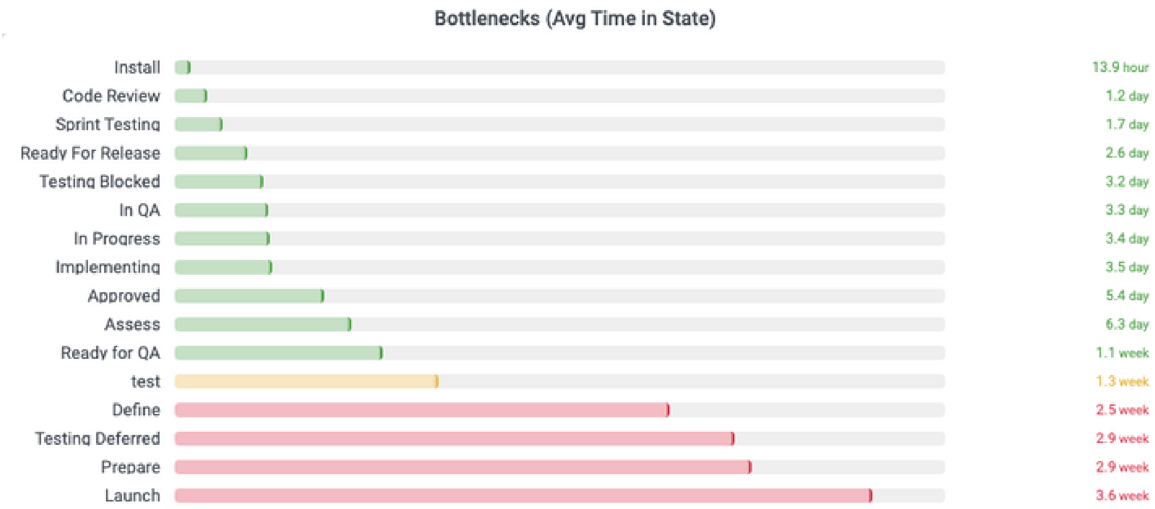
### What does this tell you?

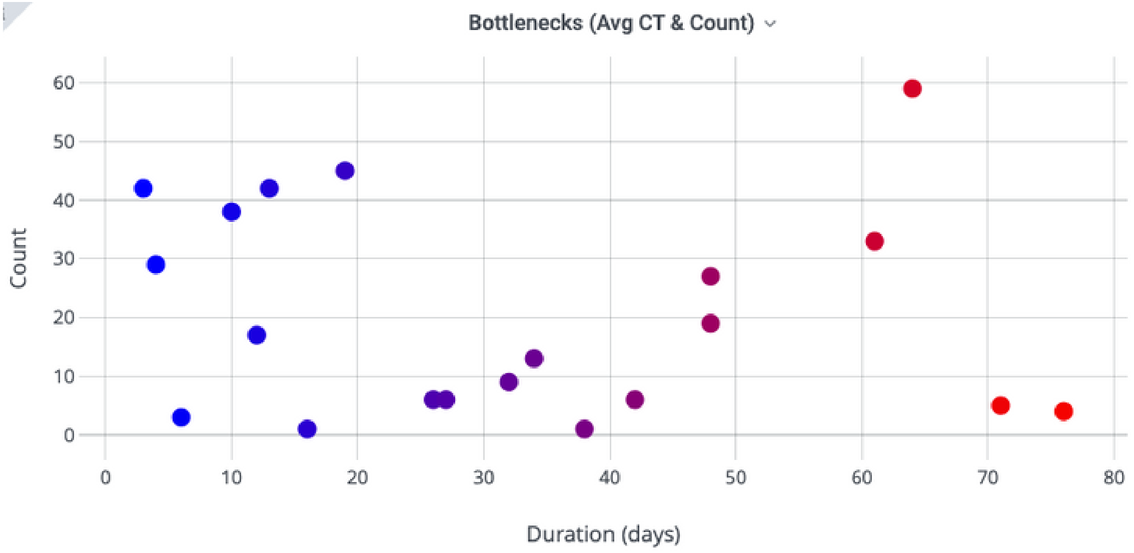
- Lets you know if there is a state in the workflow that is getting more work requests than it can process at its maximum throughput capacity.
- Causes interruption and delays across the rest of the production process.
- As a metric, this is tracking the cycle time for a given work state and determining if work is piling up at a particular stage.

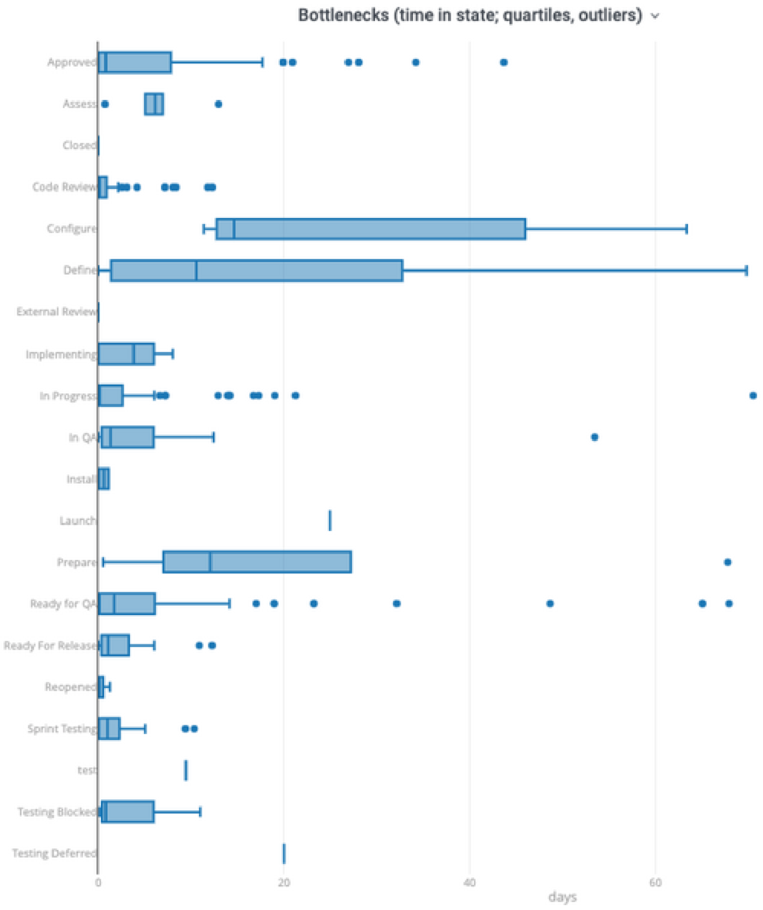
**We recommend measuring bottlenecks in 3 separate graphs: Avg Time in State, Average Cycle Time w/ Count, and Time in State w/ quartiles and outliers included*

### Why do you need it?

- Pinpoints where changes can make the biggest impact in your software delivery value stream.
- Helps you understand where you need to focus to make the value stream more efficient.
- Manage your throughput variation by assessing what is causing your bottlenecks.
- If there is a bottleneck in a particular stage, determine how you can balance cycle times across the value stream: eliminate, combine, rearrange, and/or simplify the steps.







## Why 3 bottleneck views?

Using these 3 different views of the bottleneck metric is critical to paint a complete picture.

Average time in state" is a great basic place to start, since seeing every stage stack-ranked makes it easy to identify the longest steps in your process.

However, you also need to consider the relationship between average cycle time & count (aka the volume in that stage). This way, you are able to properly prioritize your improvement efforts based on both time and number of issues.

The final chart outlines quartiles and outliers for each of your stages. This is important to give further context regarding your bottlenecks and to let you know if there are outliers skewing your numbers.

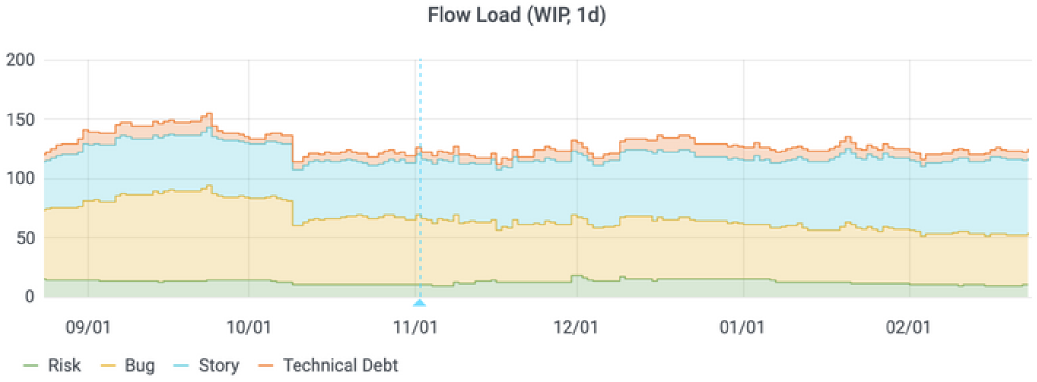
# Flow Load (WIP)

### What does this tell you?

- The WIP (work in progress) of a particular type on a particular day — the number of flow items being actively worked on in a value stream.
- Shows what exactly your team is spending time on.
- How balanced (or unbalanced) your work is across each of your categories
- Tracking Flow Load shows changes in Velocity and Time.

### Why do you need it?

- Track inefficiencies caused due to too many flow items that are reducing output.
- Maximize Velocity and minimize Time.
- Allocate your work and make your prioritisation decisions.
- Indicates if you’re spending too much effort on either fixing the product or improving it



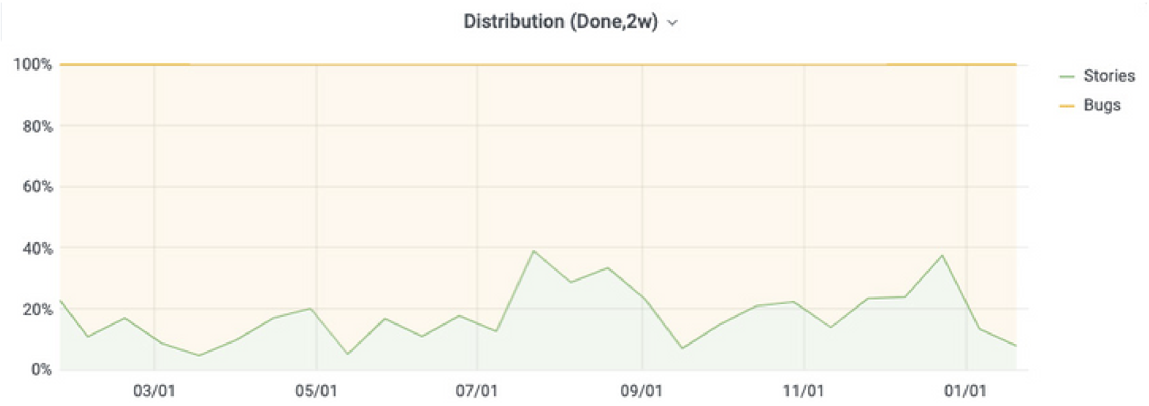
# Distribution (Done)

### What does this tell you?

- Shows how much work your team completed in a time period and the specific types of work items that are being focused on.
- Similar to flow load, except focuses on the completed work rather than WIP.
- It also gives a perspective of both the total Throughput and Throughput for individual work items over a certain time period.

### Why do you need it?

- Understand how to accelerate delivery of business value by prioritizing on completing fixes or enhancements as per business/customer needs.
- Ensure you are capable of bringing new features to customers rapidly while also maintaining the quality of the current product.
- Learn where to refine your investments over time.
- Understand your Throughput Variation by looking at a full breakdown of the completed work.



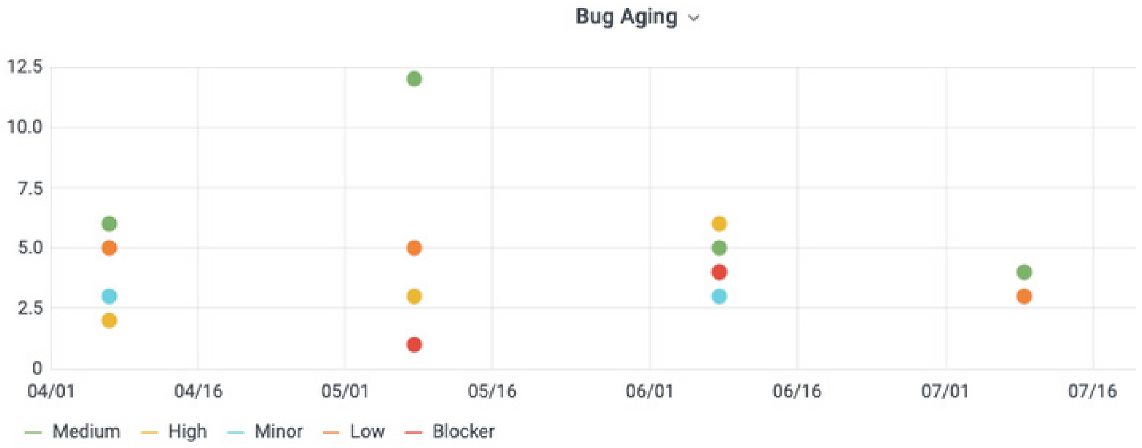
# Bug/Defect Aging and Defect Backlog

### What does this tell you?

- Shows if your team is neglecting high-priority defects in the process.
- Indicates if you have any defects that are old and of high-severity level that are not being addressed.
- If there are blockers or high-priority defects that could be negatively impacting the customer experience, either through delays or unaddressed product issues.

### Why do you need it?

- Prioritize clearing high-priority defects that will prevent critical problems in the present or the near future.
- Constantly check on the health of a software and improve customer satisfaction by ensuring major defects are fixed in a timely manner.
- Highlight process, prioritization, or quality issues if older high-priority defects are piling up.



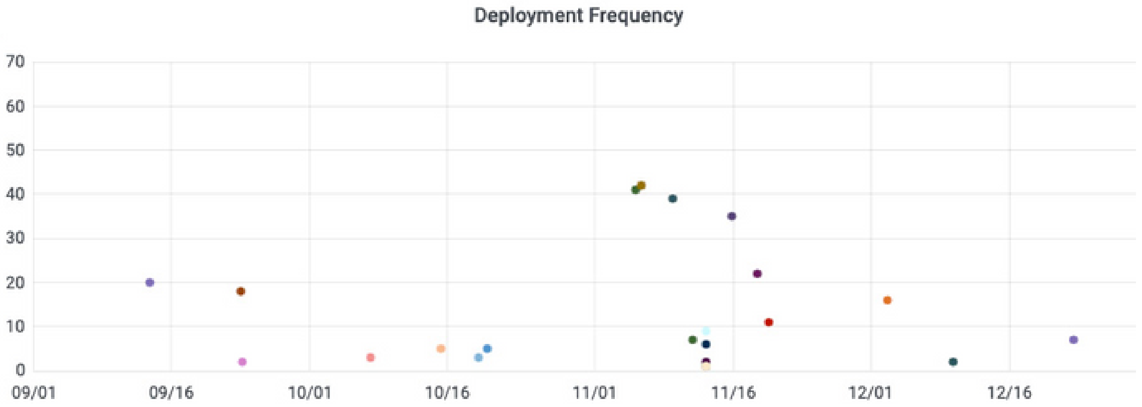
## Deployment Frequency (DF)

### What does this tell you?

- Tracks the frequency with which increments of code are deployed to staging, testing and production.
- Analyse the frequency of deployments by branch, portfolio, programme, repository or team.
- See if you are hitting the targets that have been set for frequency of deployment, or if your teams are missing the mark and causing delays.

### Why do you need it?

- Ensures the delivery of valuable software ‘early and often’ by creating a trend towards more frequent deployments and small tickets.
- Track improvements and celebrate the wins if your organization is new to Agile software delivery and looking to increase Agile DevOps maturity.
- Baseline your organization’s agility, and see if changes are increasing it.
- Increasing DF enables more effective customer feedback loops.



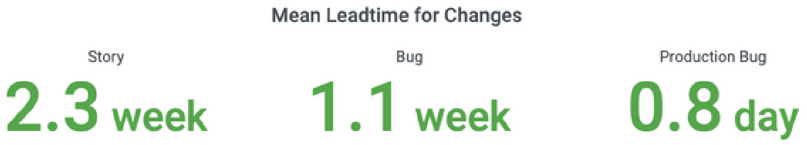
## Mean Lead Time for Changes (MLT)

### What does this tell you?

- MLT is the time taken to go from code committed to code successfully running in production.
- It is the average lead time (also known as Change Lead Time) for different types of tickets such as stories, bugs, evaluations, and improvements from idea to production.

### Why do you need it?

- Help your teams reduce their overall Lead Time and increase the velocity of software delivery.
- Keep track of bottlenecks and Throughput Variation that may be occurring after the code is committed.
- Useful to assess how quickly your team can respond to an urgent need from an internal or external stakeholder.
- Be more predictable and accurate in determining when a change is expected to be delivered.



Mean Leadtime for Changes

| Story | Bug | Production Bug |
|---|---|---|
| 2.3 week | 1.1 week | 0.8 day |

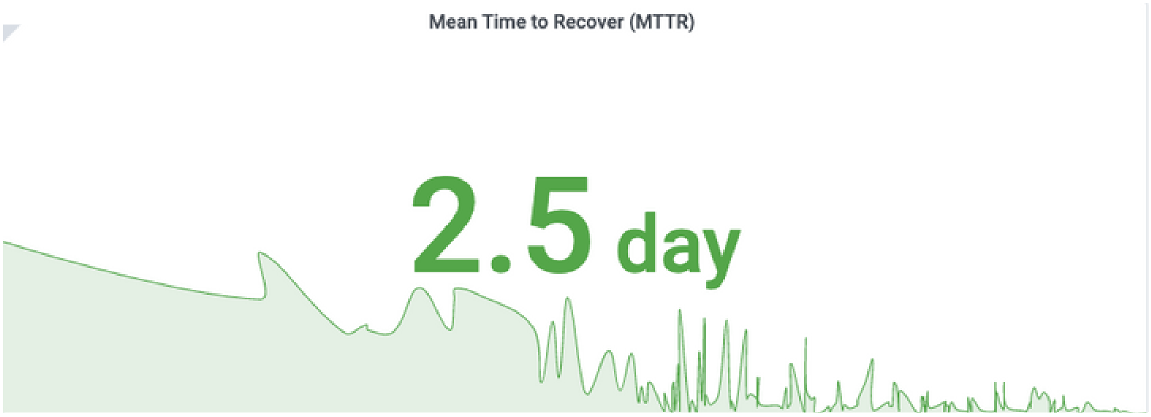
## Mean Time to Recover (MTTR) or Mean Lead Time to Recover

### What does this tell you?

- MTTR is the average time required to repair a failed component, system, service or defect that is in production.
- Identifies how long it takes for the next successful workflow on a build branch that has failed.

### Why do you need it?

- Track and manage build failure and reduce friction in deployment.
- If you are in an early stage of Agile DevOps maturity, use this to manage recovery time and improve Deployment Frequency.
- Focus on improving this metric to respond more quickly to unexpected issues.



Mean Time to Recover (MTTR)

# 80th Percentile Lead Time Expectation

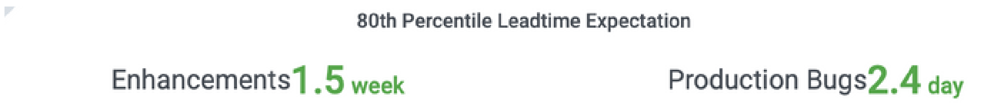
### What does this tell you?

- It is overall time to deliver an increment of software from initial idea through to deployment to live.
- Lead time for a given class of work (example, story, task, bug) is the duration between when the request was made and when the solution is available to the requestor.
- Lead time is always from the customer's or end user's perspective.
- Instead of the average lead time, the 80th percentile of your lead time indicates that 80% of the historical lead time falls below that point.

### Why do you need it?

- Plan and make forecasts. Get a higher chance to over deliver by underpromising.
- To monitor the lead time from a trend perspective.
- Create a palatable customer experience by giving a 80% probability of giving a fix in a certain period of time.
- Reduce risks of disappointment by delivering as promised.*

**We recommend 80% lead expectation rather than the average to help manage customer expectations. Based on historical data, you will meet your prediction 80% of the time, rather than half the time with a hard average.*



80th Percentile Leadtime Expectation

Enhancements 1.5 week

Production Bugs 2.4 day

**These are the core metrics that we recommend for teams to deliver software more predictably. But what about your services and support teams?**

In addition to the above metrics, ConnectALL Insights Analytics is equipped to capture specific metrics for every team and value stream Some examples for other teams include:

### For Support Teams

- Count by Severity
- Agent Load
- Created vs Resolved Tickets

### For Service Teams

- Distribution by Account Manager
- Distribution by SA
- Evaluation Resolution

You can get all these and more, including customized metrics based on your needs. Let us help you diagnose the most critical metrics for your organization based on the specific problems you need to solve.

**If you are ready to put your metrics game face on, contact our specialized sales representatives, who will guide you on the path of predictability.**