[WIP]: Core Engineering metrics

Context

This document describes all the metrics we want to track in order to monitor how Platforms teams perform, which directly connects with Platforms Engineering Q2 OKR - https://app.7geese.com/objective/2097831. This work also serves as the foundation for better explaining the impact of AI (tools and agents) on the engineering teams, starting with Platforms teams (also connected with another OKR - https://app.7geese.com/objective/2093202).

Tracking how Engineering teams are performing is crucial for any type of serious and competitive Engineering structure. Doing so allows us to better understand how we actually work and what needs to be improved over time instead of just relying on assumptions, feelings, or no numbers at all.

- **Improve predictability** – Cycle Time, Lead Time, and Throughput help teams forecast delivery dates more accurately and identify bottlenecks before they become major delays.
- **Measure flow efficiency** – These metrics reveal where work spends its time (development, reviews, QA, waiting, dependencies), enabling targeted process improvements instead of relying on assumptions.
- **Balance speed and quality** – DORA metrics ensure that increasing delivery speed does not come at the expense of reliability, by tracking deployment frequency, failures, and recovery times.
- **Support data-driven decisions** – Teams and leaders can prioritize improvements based on objective evidence rather than anecdotal feedback or isolated incidents.
- **Track improvement over time** – Consistently measuring these metrics helps teams evaluate whether process changes, tooling investments, or organizational changes are actually delivering better outcomes.
- **Increase stakeholder confidence** – Reliable delivery and operational metrics make roadmap commitments and progress reporting more transparent and trustworthy.

⚠ This is the first stage of this work, meaning that the company is not used to seeing the numbers behind the metrics defined below, making all asking questions. These questions are a sign of growth, and so engineering teams should continue to explore and be encouraged to improve over time.

Main standardization rules

Jira and ADO will be the main systems we need to rely on in order to track the metrics that are defined in the next section. But before that is described, we need to define a certain set of rules for how we approach work classification, specifically on Jira.

Definition of Done

Several metrics depend on the way we interpret work to be done. Each team should have a **clear DoD** (Definition of Done) and that should translate naturally to Jira. Usually a good DoD should be “live and enabled“ meaning the development reached the production environment and is enabled (feature toggle or AB test is switched on).

Typical states on Jira that translate to work done:

- Done
- Resolved
- Closed

> ⚠️ “Release on Stage” shouldn’t be treated as work done.
>
> “Release on Live“ shouldn’t be treated as work done if the flow itself already includes the states defined above.

Epics usage on Jira

Epics are usually used to track teams delivery, however some Product Owners are treating Epics as initiative aggregators to join all work done pre and post initiative delivery. This means that an Epic can be in progress while all work items below that Epic are done. The purpose of this document is not judge if this is good approach or not. Nonetheless, this kind of approach can destroy how an engineering team tracks delivery oriented metrics. For such reasons, the tracking tool should have an option to exclude Epics from its metric compute logic (per project).

Core metrics

Metrics will be presented in different categories, as there are different ways to look at how teams are performing. Some metrics will be more time-oriented, hence more connected with speed of delivery, while others are more related to quality and flow predictability.

> We will need to connect to Jira and/or ADO to be able to collect all the metrics that are proposed below.

Delivery & Velocity

In this category we are mostly talking about time, and it should be possible to observe all of these time-related metrics together in a single chart so it is easier to understand patterns and deviations.

Lead Time MUST HAVE

The Lead Time should measure the time between when the ticket was created until it is done. The purpose is to measure the **full end-to-end customer experience**.

Rules:

- Done aligned with Team's DoD
- Waiting time should not be excluded. Waiting time can be:
  - Non working days (Weekends, bank holidays and vacations)
  - On Hold/Blocked states
- Present time in days

Cycle Time MUST HAVE

The Cycle Time should measure the time from when the ticket is put in progress until it is done. The purpose is to measure the **actual work** that was done. Active work is usually the term used for cycle time; however, the word active gets easily misinterpreted. Cycle time is supposed to include waiting time that many times appears after it started. Effective hours/days spent on concrete engineering work without any waiting time is typically something else that's not traditional cycle time (please check [WIP]: Core Engineering metrics | Active Work Time nice to have).

Rules:

- Same rules as Lead Time
- In Jira, active work usually starts when setting a ticket to the next states:
  - In Progress
  - Any other?
- If the ticket goes back to New, the time it remains in New should not be included when the ticket goes past In Progress again.

Active Work Time NICE TO HAVE

Active Work Time tracks only when the **team is actively working**, removing all waiting time from the cycle time. It accurately reflects the work required by the Engineering teams.

Besides the waiting time already covered on the Cycle Time and Lead Time there are 2 other layers of waiting time that happen every day.

State transitions

All Jira statuses and how they are meant to be used either represent active time, passive time (wait time), or a mix of both. Examples:

- In Progress - mostly active
- Code Review - a mix of active and passive. Most of the time, moving a ticket for review means waiting on someone else to review a PR. The active work on that state is the actual time each engineer spent reviewing the PR. AI can reduce the passive time and increase the active time on this state but only if the actual time spent on a review is reduced.
- On Hold/Blocked - passive time as the team can't proceed, but this is already covered on what needs to be excluded from Cycle Time
- Ready To Test - passive time as this is usually a signal that some development is ready for something or someone to test; there is no active time here.
- In Testing - this is similar to In Progress and is mostly active time
- Ready to merge - this is mostly passive. It's true that there is an effort (active time) in doing the actual merge and resolving conflicts, but normally it's expected that this is residual active time.
- Release on Stage - mix of active and passive
- Release on Live - mix of active and passive

Working hours

The Engineering teams don't work 24/7 and per definition this metric already excludes non-working days and should also exclude non-working hours on working days, typically 8 or 9 hours of work. We are currently not tracking the time spent on each ticket in each working day. It's possible that someone is actually working on multiple tickets or even involved in meetings and other activities that aren't trackable or not worth it. So this is an attempt of standardizing 8/9 hours of work per day.

Rules:

- Done following the same rules as Lead Time and Cycle Time
- All waiting time defined in Lead Time and Cycle Time should be excluded

    - Vacations can potentially benefit from an integration with Bamboo HR
    - On Hold/Block assume such states need to be mapped
- Exclude the following states defined in the State Transitions section
    - On Hold/Blocked - already covered on the previous point
    - Failed on QA
    - Ready To Test
    - Ready To Merge
    - Any other?
- Implement a logic to comply with the Working Hours section so that non-working hours are excluded. This doesn't need to be perfect, but at least on cycle times of multiple days, it will allow removing most of the time when people are not actually working. Working Hours
- Present time in days

Lead Time for Changes (DORA) NICE TO HAVE

Lead Time for Changes is a core DevOps Research and Assessment metric that measures the time it takes for a **committed code change to successfully run in production**. The intent is to measure the actual responsiveness of our delivery system, not just the active engineering effort.

Essentially the Lead Time for Changes will behave like the Cycle time, the only difference is that it starts to count when an associated commit is pushed instead of the ticket being set to In Progress.

Rules:

- Same rules as Cycle Time

Time To First PR NICE TO HAVE AI DRIVEN

The Time to First PR measures the time between when a **ticket was put in progress until an associated PR is opened**. The intention is to measure how quickly engineers can turn a ticket into a reviewable implementation.

Rules:

- Same rules as Cycle Time

Status Breakdown NICE TO HAVE

Break down all Jira statuses from In Progress until it is done. This allows a clear picture of the time spent in all statuses used on the team's workflow, pushing for:

- Continuously reassess the team's workflow

    - Does it reflect what the team actually does?
    - Does it “hide” patterns that can be further optimized?
- Clear understanding of the exact states where the team is losing more time
- Clear optimization path on which states are the biggest offenders

Quality & Reliability

This category looks into the quality of deliveries, stability and reliability.

Prod Bug Rate MUST HAVE

The Prod Bug Rate measures the number of bugs that were **reported in the production environment**. On top of it, it should also break down this number per severity. These bugs can be created by the Incident Management process or any team member. The important part is that it should account for production bugs.

This metric should also offer a way to look at the number of bugs in a rate fashion, like the average number of bugs per week.

Pre-Prod Bug Rate NICE TO HAVE

This is essentially the same as the Prod Bug Rate but for all bugs found on **non-productive** environments.

Change Failure Rate MUST HAVE

Change Failure Rate (CFR) is a core DORA metric that measures the **percentage of software deployments to production that result in a degraded service**, impairment, or outage and subsequently require immediate remediation. It evaluates the quality and stability of your delivery pipeline, ensuring that speed (measured by Lead Time) does not come at the expense of stability.

A deployment is a failure if it requires:

- An emergency **rollback** to a previous version.
- A **hotfix** or urgent "fix-forward" patch shipped outside the normal cadence, for example, a major incident (P0 or P1) triggered by the Incident Management process but only if the cause was effectively from the code that was shipped.

Rules:

- Change Failure Rate is a percentage metric calculated as the (number of failed deployments / total number of production deployments) x 100
- Failed deployments obtained as mentioned before

Flow & Predictability

This category looks into how predictable teams and their flow can be.

Throughput MUST HAVE AI DRIVEN

The Throughput should measure the number of completed tickets within a timeframe as well as the average number of completed tickets per week. Usually the latter is more useful to explain what's the usual flow of the team on a week basis.

Rules:

- If Epics are excluded from delivery, they should not count for the throughput. Please check section [WIP]: Core Engineering metrics | Epics usage on Jira

Deployment Frequency MUST HAVE AI DRIVEN

The Deployment Frequency measures **how often a team/project successfully releases code to the production environment**. It serves as a primary indicator of the team's overall throughput, agility and batch size. Like Throughput it should show the total number of successful production deployments for a given timeframe as well as a rate basis. There should be 3 types of rate:

- Daily
- Weekly
- Monthly

Planned vs Unplanned Work MUST HAVE

The Planned vs Unplanned Work **measures how the flow of a team can be impacted by unexpected work**. The distinction between planned and unplanned work separates value-generating progress from unpredictable operational friction. Unplanned work is any unexpected task, disruption, or emergency that breaks the current iteration or sprint focus. It represents the operational tax paid for system instability, technical debt, or shifting priorities.

Maintenance vs Unplanned Work

Usually, teams should reserve capacity for maintenance work (per sprint, per week), which means that if there is unplanned work, like an urgent issue to fix, this should be treated as planned work if the effort falls under the reserved capacity. If more capacity is needed for the same or other tasks, then it becomes Unplanned work.

Rules:

- If the team does not make any capacity planning for maintenance, any kind of deviation, anything that wasn't planned, should be interpreted as unplanned work (bugs, support requests, stakeholders requests, ...). Teams following this approach should carefully assess how realistic their planning and deliveries can be in the constant presence of significant amounts of unplanned work. For this scenario, capture unplanned work by:
  - Status Bug, Incident
  - Label: unplanned_work
- If the team reserves capacity for maintenance, then unplanned work should only be accountable after the reserved capacity has been maxed out. For this scenario, capture unplanned work by:
  - Label: unplanned_work

PR Throughput NICE TO HAVE AI DRIVEN

The PR Throughput should measure the number of merged PRs within a timeframe as well as the average number of merged PR per week.

PR Acceptance Rate NICE TO HAVE AI DRIVEN

The PR Acceptance rate should measure the percentages of merged PRs vs the ones that were abandoned.

AI Participation Rate NICE TO HAVE AI DRIVEN

The AI Participation Rate rate should measure the percentages of tickets done that had an AI Agent Assigned.