Plandek

Applying the DORA Metrics to Drive DevOps Success

Whitepaper

Contents

About Plandek

Plandek is an intelligent analytics platform that helps software engineering teams deliver value faster and more predictably.

Celebrated by Gartner and Forrester as a 'leading global vendor', Plandek mines data from delivery teams' toolsets and gives them the opportunity to optimise their delivery process using both intelligent insights and predictive analytics.

Co-founded in 2017 by Dan Lee (founder of Globrix) and Charlie Ponsonby (founder of Simplifydigital), Plandek is based in London and currently services the UK, Europe, the Middle East and North America.

Contact: Senior Account Executive Sam Kange, skange@plandek.com

1.

DORA Metrics Defined

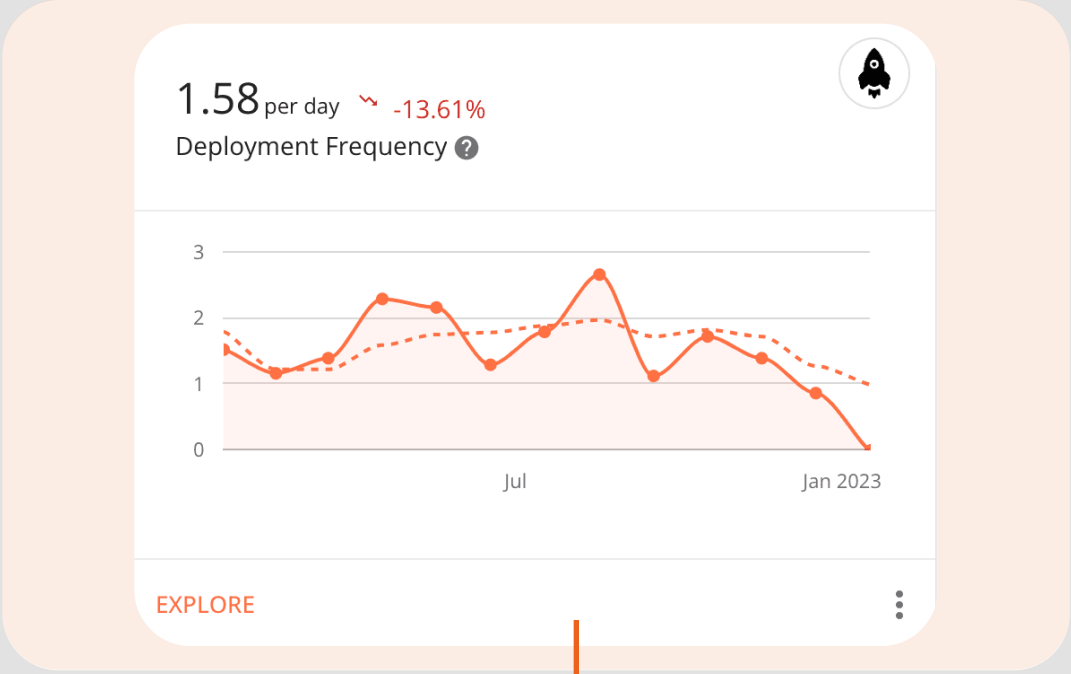
The four DORA metrics were conceived and used by Google's **DevOps Research and Assessments** team. They are the outcome of several years' research into software engineering team effectiveness.

The DORA metrics are the four key DevOps metrics that DevOps teams can measure to define their performance against benchmarks ranging from 'low performers' to 'elite performers'.

DORA Metric	Definition
Deployment Frequency (also called: DF, deploy frequency)	The frequency at which new releases deploy to production
Change Failure Rate (also called: CFR, change fail percentage)	The percentage of deployments causing a failure in production
Lead Time for Changes	The time between commit to production
Mean time to Restore Service (also called: MTTR, mean time to recover	How long it takes an organization to recover from a failure in production

Deployment Frequency

How often does your organisation deploy code to production or release it to end-users?



Deployment Frequency
Plandek DORA Metrics Dashboard

Deployment Frequency is a core DevOps metric and more broadly a core Agile delivery metric.

As the name suggests it tracks the frequency with which increments of code are deployed to production.

As the core objective of Agile software delivery is '...the early and continuous delivery of valuable software..', Deployment Frequency is a core Agile metric and represents a core objective of effective DevOps.

The software delivery process should be seen as an end-to-end value delivery process – as such Deployment Frequency is best viewed alongside a broader range of agile delivery metrics measuring value delivery, delivery efficiency, dependability, backlog health, delivery and engineering quality, and DevOps processes.

The calculation of Deployment Frequency requires surfacing data from CI/CD tools (e.g. Jenkins, CircleCI, Github Actions).

This is typically done via reporting plug-ins to such tools or via an end-to-end delivery metrics dashboard, such as Plandek.

Why does Deployment Frequency matter?

Deployment Frequency is a key DevOps metric used to ensure that software is delivered early and often.

Emphasis might be placed on tracking and improving development-oriented metrics such as Cycle Time and Throughput – but the acid test is that the software developed by the engineering team is regularly deployed to live (otherwise all that effort is wasted).

Hence Deployment Frequency is not only a critical DevOps metric, but also a critical broader Agile delivery metric.

Indeed, there is a strong body of research which correlates improved customer NPS (Net Promoter Scores), with higher levels of Deployment Frequency.

This is logical as companies that can most rapidly respond to customer needs and provide new features most regularly, will over time better satisfy their customers.

Plandek users see

up to

400% increase

in Deployment Frequency

> 'Leaders combine gut instincts with analysis to make more informed decisions.'

'The State of Value Stream Management' Trend Report
Forrester Research Inc., 2022

Furthermore, experience shows that high frequency deployments optimises value delivery for **three practical reasons**:

1. Smaller release size means easier testing and deployment – so Change Failure Rate and Mean Time to Recover fall and availability rises.

2. More regular releases enable a more effective customer feedback loop so that developers can respond to customer needs faster and reduce time spent building unwanted features.

3. Engineers feel closer to the customer and hence work more effectively.

Change Failure Rate

What percentage of changes to production or end-users results in degraded service?

Change Failure Rate
Plandek DORA Metrics Dashboard

Change Failure Rate is a core DevOps metric, and more broadly a core Agile delivery metric.

Change Failure Rate is an extremely helpful metric which identifies the percentage of workflows which fail to enter production, as well as the overall risk that this poses to development.

All interruptions pose a significant risk to the SDLC, both in day-to-day development and responding to incidents due to the delays.

Tracking all your workflows over a set period of time, Change Failure Rate calculates the percentage of those workflows that ended in failure or required remediation (e.g., require a hotfix, rollback, fix forward, patch).

Plandek's powerful filtering enables Change Failure Rate analysis by a number of different dimensions – such as branch, pipeline, portfolio, programme, repository and team – in order to reduce it.

Reducing Change Failure Rate will reduce overall Lead Time and increase software delivery velocity and quality.

Why does Change Failure Rate matter?

Change Failure Rate is a very useful DevOps metric to help teams reduce their overall Lead Time.

Additionally, Change increases the velocity of software delivery and minimises negative impacts of failure rates upon users.

Deployment failures are a key source of friction in the end-toend delivery process and waste time and resource – hence the focus on reducing the Failure Rate.

Change Failure Rate is particularly important for organisations looking to increase delivery velocity and reduce Lead Time.

Specifically, Change Failure Rate is essential for:

1. Delivery organisations at an early stage of Agile DevOps maturity.

2. Large-scale delivery capabilities with distributed teams (onshore, offshore, contractor, inhouse) and potentially a higher turnover of engineering talent.

3. Teams involved in strategically critical software delivery projects.

Plandek users see

up to

54% reduction

in Escaped Defects

Official Case Study: Plandek x Pandora

PANDORA

'Having Plandek's LiveView and being able to know where I need to spend time [and] where I don't need to spend time, it's been very, very useful.

There is no other tool that would allow me to be able to do that.'

Jacob Harrison
Scrum Master, Pandora

Lead Time for Changes

How long does it take to go from code committed to code successfully running in production?

Lead Time for Changes
Plandek DORA Metrics Dashboard

Lead Time for Changes is defined as **Accelerate: The Science of Lean Software and DevOps** as 'the time taken to go from code committed to code successfully running in production'.

As such, it is very similar to the Code Cycle Time metric.

Code Cycle Time is a broader metric in that it provides insight into the different stages that a Pull Request goes through as well as the time to deploy.

These stages are defined as:

1. Time to Review: From open to the first comment or review
2. Time to Approve: From the previous stage to approved
3. Time to Merge (Commit)/Close: From the previous stage to merge/close
4. Time to Deploy: From the previous stage to deployed to production.

Given these stages of Code Cycle Time, it's essential to understand that Lead Time for Changes focuses only on stages 3 and 4.

Why does Lead Time for Changes matter?

Lead Time for Changes is a very useful DevOps metric to help teams reduce their overall Lead Time and increase the velocity of software delivery.

The Pull Request process can become a key blocker in the overall delivery process as code awaits peer review.

This may be because teams rely on a few key engineers to action the majority of Pull Requests and hence bottlenecks quickly develop.

Empirical data shows that Lead Time for Changes typically accounts for around 30% of Cycle Time. This means that the Pull Request process is often a key source of 'hidden' delay.

'Our intelligent analytics helps software delivery teams deliver value faster and more predictably as Plandek continues to lead the data science revolution in DevOps.'

Charlie Ponsonby
Co-founder and CEO, Plandek

Plandek users see

in Time to Value

Lead Time for Changes is particularly important for organisations looking to increase delivery velocity and reduce Lead Time.

Specifically, Lead Time for Changes is essential for:

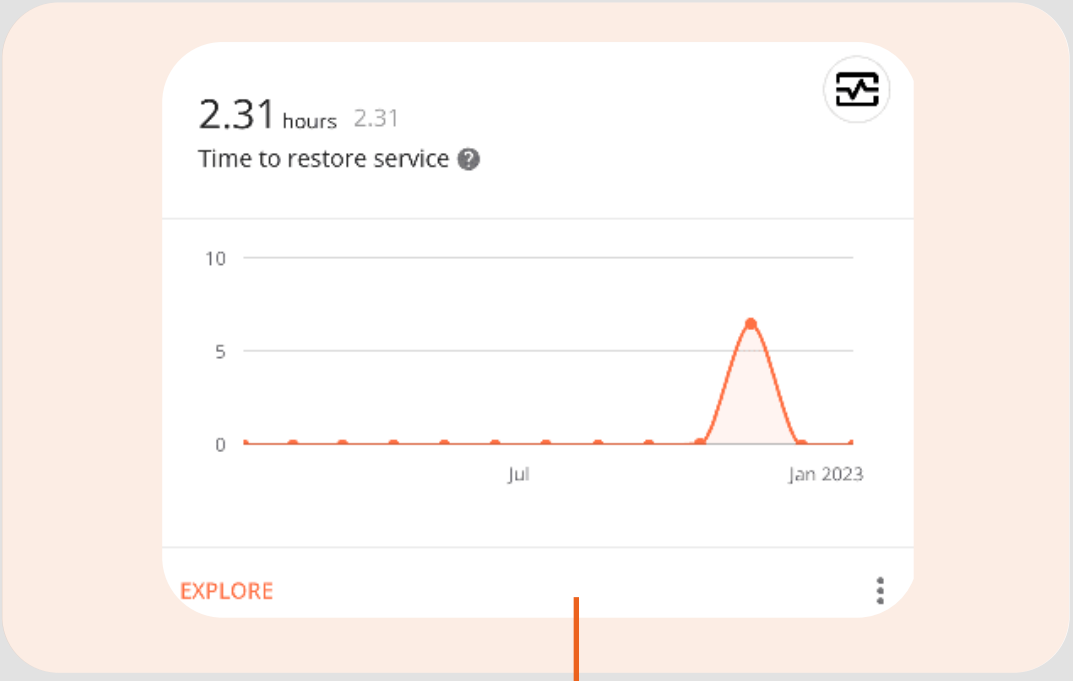
1. Delivery organisations at an early stage of Agile DevOps maturity.

2. Large-scale delivery capabilities with distributed teams (onshore, offshore, contractor, inhouse) and potentially a higher turnover of engineering talent.

3. Teams involved in strategically critical software delivery projects.

Mean Time to Restore

What is the mean length of time your organisation takes to restore service following an incident?



Mean Time to Restore
Plandek DORA Metrics Dashboard

Mean Time to Restore (MTTR) measures the average time it takes for an organisation to recover from a systems outage.

This is integra information, which means MTTR is a key DevOps KPI – as well as being closely linked to many Agile metrics.

It is a critical measure of end-customer experience. As such, it is essential that organisations track and manage Mean Time to Restore as part of a broader set of customer experience metrics.

Mean Time to Recover (MTTR) is calculated by adding up all the systems downtime over a period and dividing that time by the number of incidents.

$$\frac{\text{System downtime}}{\text{Number of incidents}} = \text{MTTR}$$

For example: if your systems were unavailable for 60 minutes over a 24 hours period across two separate incidents, your MTTR would be 30 minutes.

Why does Mean Time to Restore matter?

MTTR is calculated using data from monitoring tools like PagerDuty, NewRelic, Splunk or Dynatrace.

MTTR is a high-level metric that helps you identify if you have a problem with recovery.

As such, it cannot be used by itself to diagnose where the problem lies within your software delivery and DevOps processes. There are many interdependencies between failure and recovery, including:

1. **Alert systems**: is there a delay between a failure and an alert?

2. **Diagnostics**: Are we able to identify the root cause of the problem?

3. **Repairs**: is the repair/bug fixing undertaken as efficiently as possible?

Plandek users see

in Flow Efficiency

This is where end-to-end software delivery metrics platforms like Plandek play such an important role.

Plandek surfaces the critical secondary DevOps metrics, Agile delivery metrics and engineering metrics that give tech leaders valuable insight into the reasons for failure and the MTTR.

Official Case Study: Plandek x Hg Capital

'Plandek enables our portfolio companies to track and drive their R&D ROI, as part of their broader value creation approach.

The effectiveness of their technology delivery is a key piece of that jigsaw.'

Stuart Pearce
Portfolio CTO, Hg Capital

2.

Surfacing Accurate DORA Metrics

The DORA metrics are derived from a number of data sources (see below).

DORA Metric	Data source
Deployment Frequency (also called: DF, deploy frequency)	CI/CD tools such as Jenkins and CircleCI
Change Failure Rate (also called: CFR, change fail percentage)	Service Management and/or CI/CD tools
Lead Time for Changes	Repository & CI/CD tools
Mean time to Restore Service (also called: MTTR, mean time to recover	Service Monitoring tools

To be of real benefit, a reasonably sophisticated BI solution is required, which enables:

1. Surfacing of DORA metrics in real-time with no effort.

2. Organisation-wide visibility of relevant metrics.

3. Drill-downs and filters to understand metric behaviours.

4. Surfacing of second-order (determinant) metrics that directly impact the DORA metrics.

5. Team/squad visibility of metrics, meaning metrics are 'owned' at all levels.

However, these requirements are tricky to enable without a specialist (and stack agnostic) Agile DevOps metrics and analytics platform like Plandek.

Plandek allows you to:

1. View data at any organisational level.

2. Apply filters and breakdowns to view detailed metrics based on your own criteria.

3. Show and compare time series.

4. Overlay related DevOps and delivery metrics to identify correlations.

3.

Second-order DevOps Metrics That Help Drive Improvement in DORA

The end-to-end software delivery process is complex, hence there are a great number of 'second order' (determinant) Agile delivery, DevOps and engineering metrics that require consideration in order to drive significant movement in the DORA metrics themselves.

Experience within delivery teams amongst our clients globally suggests that the second-order metrics presented in the graphic below are incredibly helpful when looking to successfully implement the DORA metrics.

DORA Metrics	Second Order (Determinant) Metrics
Deployment Frequency	Cycle Time Flow Efficiency Mean Time to Merge PRs Developer to Pipeline Ratio
Lead Time for Changes and Change Failure Rate	Mean Time to Merge PRs Open PR Age Flakiest File Analysis Escaped Priority Defects (e.g. P1,P2)
Mean Time to Restore	Incident Resolution Time Incident Cluster Analysis

CT
Cycle Time

Second-order metric definition

Cycle Time is a core agile software delivery metric which impacts an organisation's ability to delivery software early and often. Cycle Time looks at the development time elapsed and time spent in each status within the development process, in order to identify bottlenecks. Identification and removal of these bottlenecks (e.g. code awaiting QA), reduces the time taken to deliver an increment of software and therefore provides the opportunity to increase Deployment Frequency (as you can't deploy code until it's built!).

FE
Flow Efficiency

Second-order metric definition

Flow efficiency is a critical measure of delivery efficiency that all delivery teams should track and act upon. It analyses ticket statuses to calculate the proportion of time tickets remain in an 'active' status versus an 'inactive' status.

As such, it shows a helpful overall measure of the efficiency of your end-to-end delivery process. It is not uncommon for Flow Efficiency percentages to be as low as 10% - hence for 90% of the total Lead Time, tickets are sitting in an 'inactive' status (such as 'awaiting' or 'queuing' at a certain point in the process). Improved Flow Efficiency therefore ultimately enables increased Deployment Frequency.

MTTM
Mean Time to Merge PRs

Second-order metric definition

The PR process is often a key blocker to increasing velocity as it may account for up to 30% of total Lead Time. Optimising this element of the SDLC is therefore another important enabler of increased Deployment Frequency.

Mean Time to Merge PRs looks at all your completed Pull Requests (e.g. closed, merged, declined etc) within the specified time range and shows the average hours to complete, from when the PR was opened. Not only that but it provides full insight into the different stages that a PR goes through.

DTPR
Developer to Pipeline Ratio

Second-order metric definition

Architecture is a critical enabler of increased Deployment Frequency. The trend towards smaller applications and microservice architecture is in large part designed to enable increased Deployment Frequency.

A neat determinant metric in this regard is therefore Developer to Pipeline Ratio, as a high ratio of developers to a pipeline indicates large complex applications and a likely inability to significantly increase Deployment Frequency.

OPRA
Open PR Age

Second-order metric definition

As the name suggests, Open Pull Request Age tracks the average age of open PRs. It can be reviewed down to the individual level and gives a good indicator of the effectiveness of the PR process, which itself is an important determinant of Lead Time for Changes.

FF
Flakiest Files

Second-order metric definition

Flakiest Files (which is only available in the Plandek dashboard) correlates commit and build data to identify the files that are the source of build failure. This enables teams to quickly find flaky files and resolve issues more effectively.

EPD

Escaped Priority Defects

Second-order metric definition

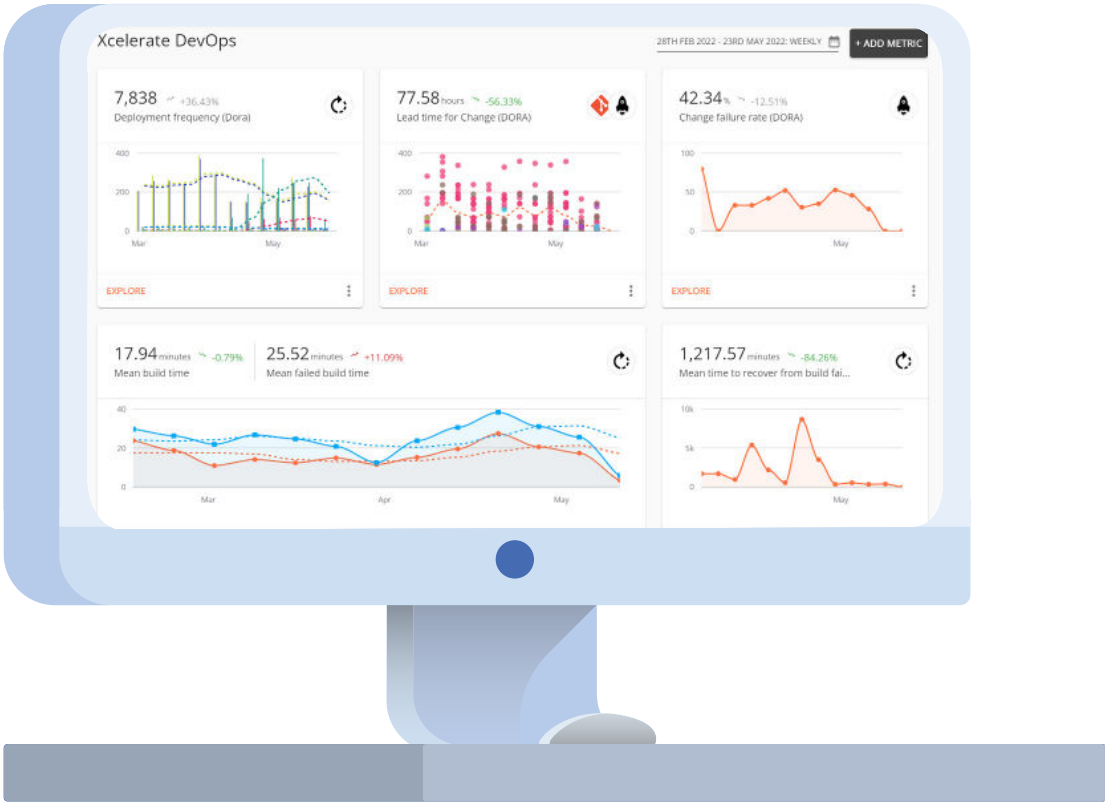
Change Failure Rate represents the percentage of deployments causing a failure in production. Change Failure Rates are typically low and difficult to pre-empt. An interesting related metric is therefore Escaped Defects as an Escaped Defect is a defect that has similarly made it all the way into production and should have been caught in the development process during testing. Escaped Defects can be analysed by branch, pipeline, repo, team and more to provide valuable insight.

IRT/ICA

Incident Recruitment Time and Incident Cluster Analysis

Second-order metric definition

Incident Cluster Analysis is helpful to identify the source of incidents and hence the likely rootcause, with a view to reducing the Mean Time to Restore. Incident Resolution Time can be analysed by type of incident and by source in order to more fully understand the nature of the incidents themselves and how to reduce future incidents and, thereby, reduce Mean Time to Restore.



An analytics platform like Plandek enables organisations to surface any of these determinant metrics in real time, at any level within the organisation.

The visual above shows an example of a fully-customisable Plandek dashboard.
In this case, it has been created to focus on a range of second-order DevOps metrics.

4.

Implementing the DORA Metrics Across Teams

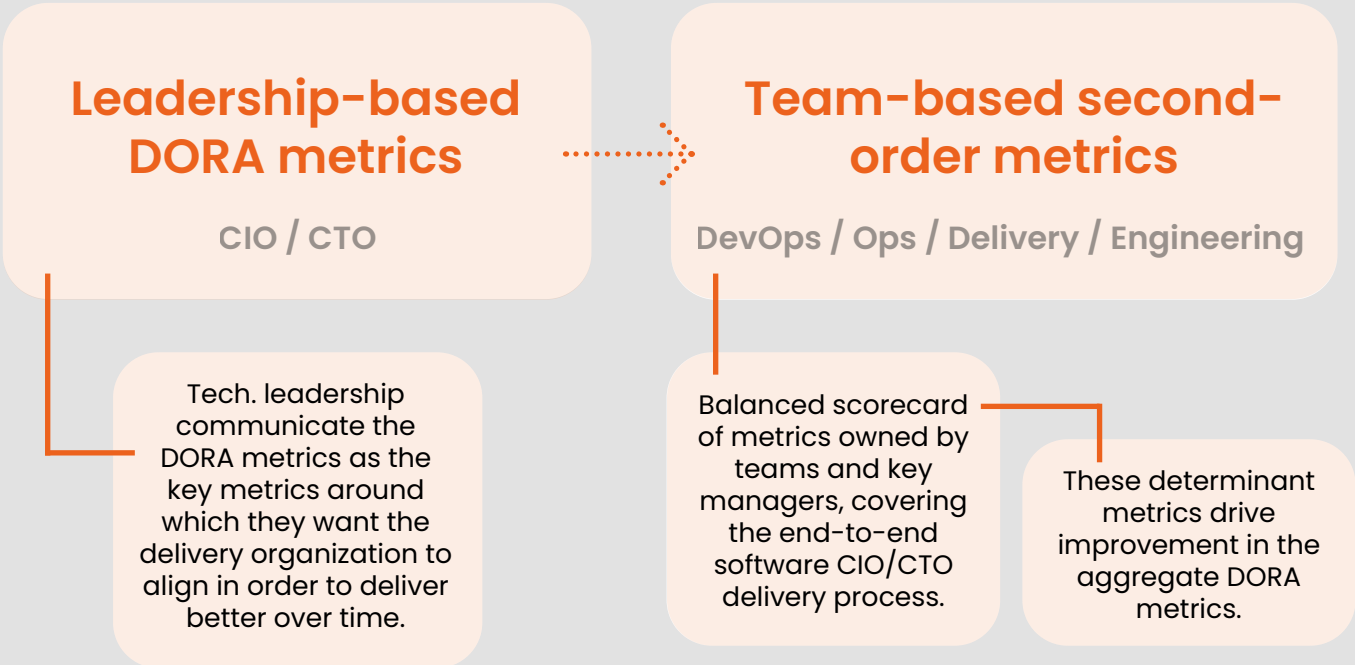
Insight-driven Agile Delivery© (IDAD) is a framework that can be easily applied to help implement the DORA metrics. It helps you identify the most relevant 'second-order' metrics and provides a framework to cascade them out across the delivery organisation, so that end-to-end delivery metrics are understood and adopted by everyone.

As shown conceptually below, the metrics are selfreinforcing, with team and manager level metrics rolling-up to the overall DORA metrics sponsored by the technology leadership team.

The second-order metrics are cascaded across the delivery organisation, with customised dashboards provided (using a tool such as Plandek) to surface a balanced scorecard of metrics owned by teams and key managers, that cover the end-to-end software delivery process.

The key is that the team-level metrics are adopted enthusiastically by the teams themselves and are deterministic of improving the DORA metrics at aggregate level.

IDAD framework to underpin your implementation of the DORA metrics:



In order to drive improvement over time, it is key that the metrics are embedded into team culture and process. Embedding this kind of continuous improvement programme across a busy delivery organisation is not a trivial task.

It requires:

1. Strong sponsorship from the leadership team.
2. A well-defined shared hierarchy of metrics (DORA and second-order metrics) which have been agreed with full involvement of Team Leads.
3. A means of surfacing those metrics in customisable dashboards available at all levels within the delivery organisation.
4. Strong sponsorship and ownership at Scrum Master and Team level so Team Leads embed metrics in the 'rhythm' of their workflow, so that the metrics are embedded in their daily, weekly and sprint processes.
5. Targets are set and where appropriate these targets are reflected in incentive structures so that teams 'own' the metrics and embed them in their stand-ups, sprint retros and other key communication points.

When these conditions are met, it enables a virtuous cycle of team-driven continuous-improvement that quite quickly translates into significant and lasting delivery performance improvements reflected in the DORA metrics at the organisational level.

Contact Plandek

Want to learn more about Plandek?

Sign up for a free account today to explore the platform or book a custom demo with our team.