Copyright Plandek – January 2021

Metrics to track and drive your Agile DevOps maturity

This short whitepaper considers sensible metrics for 'Early-stage', 'Intermediate' and 'Advanced' level Agile DevOps practitioners, so organisations can track their progress over time and self-improve around sensible metrics suited to the maturity of their Agile DevOps capability.

At Plandek we specialise in end-to-end software delivery metrics and analytics. As such we work with clients of all shapes and sizes – all of which are on some form of (Agile) transformational journey.

Advanced or 'Mature' Agile practitioners tend to be many months into their Agile transformation, with well established Agile methodologies (often at scale); an effective set of DevOps tooling; Delivery and Operations teams aligned to deliver in an Agile way; and business stakeholders who understand Agile principles. As a result, these 'mature' Agile businesses are highly proficient at delivering quality software dependably, early and often.

At the other end of the spectrum are those organisations very early in their Agile transformation – who are just starting to implement an Agile way of working across their delivery teams, and are at the early stages of implementing the DevOps toolsets to underpin those processes. Much of their software delivery may still be 'waterfall' in nature and they may not have yet achieved an effective CI/CD (continuous integration/continuous deployment) methodology.

It stands to reason that effective measurement of the end-to-end Agile delivery process is critical in order to track and drive improvement over time – and ultimately to deliver the core agile goal of "the early and continuous delivery of valuable software" (Source: Agile Manifesto). As such, there are many agile, delivery and engineering metrics that can be used. **But how do you choose sensible metrics for your organisation's level of Agile maturity?**

This short paper considers our recommended metrics for mature, intermediate and early-stage Agile delivery practitioners. It considers these metrics in a hierarchical way – with suitable metrics for technology leadership, delivery & engineering management, and the teams themselves.

Using metrics to understand the health of your delivery capability and your DevOps maturity

Agile software delivery is a complex process that is very often hiding very significant inefficiencies and bottlenecks.

Fortunately the process is easily measureable as there is a rich digital footprint in the tool-sets used across the process – from pre-development; development; integration & deployment; and out into live software management.

However surfacing data from these myriad data sources (toolsets) and synthesising meaningful metrics that compare 'apples with apples' across complex Agile delivery environments is very tricky.

Hence until recently, software delivery metrics have been much discussed but little used.

This has changed very significantly with the arrival of BI solutions like Plandek that enable the surfacing of accurate end-to-end software delivery metrics for the first time. As a result the field is moving centre stage, with Gartner and Forrester (to name but two examples) starting to advocate the importance of metrics and analytics in effective Devops Value Stream Management (**See Gartner Sept 2020 Market Guide 'DevOps Value Stream Management Platforms').**

Indeed, the end-to-end view provided by Plandek, enable clients to very closely track the effectiveness and maturity of their Agile DevOps transformation.

Metrics for early-stage Agile DevOps practitioners

Plandek can surface a myriad of metrics – with dashboards customisable by users so that Team Leads, Managers and Leadership have their own dashboards reflecting their individual responsibilities and goals.

Here we select a few simple metrics that in our experience are ideal for organisations at the early stage of an Agile DevOps transformation. The metrics focus on simple measures that underpin the fundamentals of increased agility and are relatively easy to understand and act upon.

Figure 1 - Early-stage Agile Devops Maturity metrics

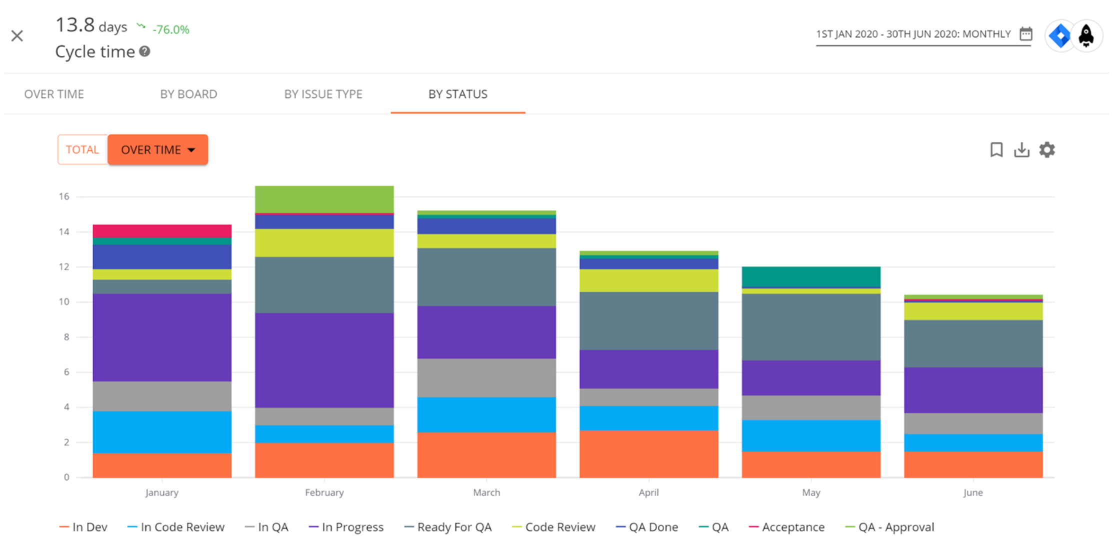
Early-stage DevOps Maturity Metrics	
1. Cycle Time	Early and continuous delivery
2. Deployment Frequency	
3. Throughput (Delivered Story Points or Value Points)	Delivery of value
4. Escaped defects	Quality of delivery

Early-stage Agile DevOps maturity metrics - Cycle Time

Cycle Time is an ideal delivery metric for early stage practitioners. It simply measures the time taken to develop an increment of software. Unlike the more comprehensive measure of Lead Time (which measures the length of the entire end-to-end delivery process), Cycle Time is easier to measure as it looks only at the time taken (within a scrum team) to take a ticket from the backlog, code and test that ticket – in preparation for integration and deployment to live.

As per figure 1 below, the Cycle Time metric view allows teams to understand time spent in each ticket status within the development cycle. Plandek has flexible analytics capability and powerful filtering to allow analysis by Status, Issue Type, Epic (and any other standard or custom ticket field) all plotted over any time range required.

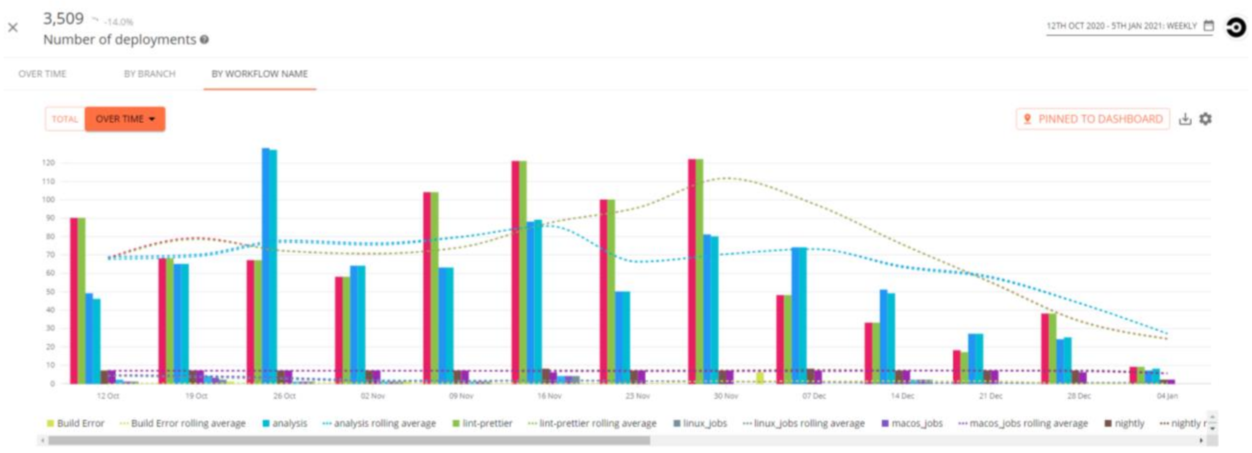
Figure 2 – Example Plandek Cycle Time metric view



Early-stage Agile DevOps maturity metrics – Deployment frequency

Deployment Frequency is another fundamental measure of an organisation's agility (when viewed alongside the other critical metrics described here). A core objective of Agile delivery is the ability to develop and **deploy to live** small software increments rapidly. Deployment Frequency tracks that basis competence and is a powerful metrics around which to focus effort at all levels in the delivery organisation at the early stages of an Agile transformation.

Figure 3 – Example Plandek Deployment Frequency metric view

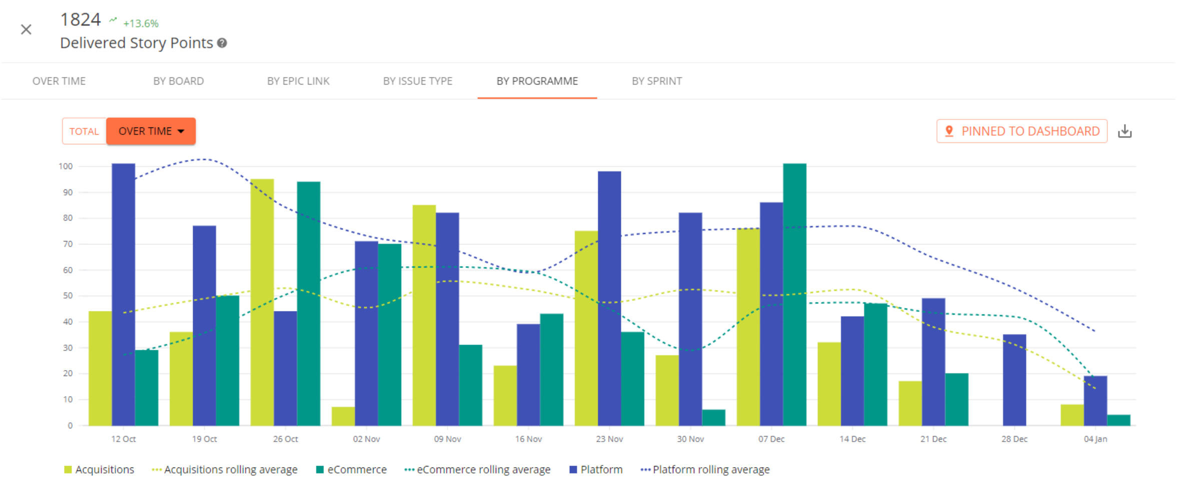


Early-stage Agile DevOps maturity metrics – Delivered Story Points

Delivered Story Points is often considered a problematic metric due to the potential inconsistencies in the calculation of story points and how much effort they represent. However, as a basic measure of output and how that is changing over time, it is a powerful metric around which to align.

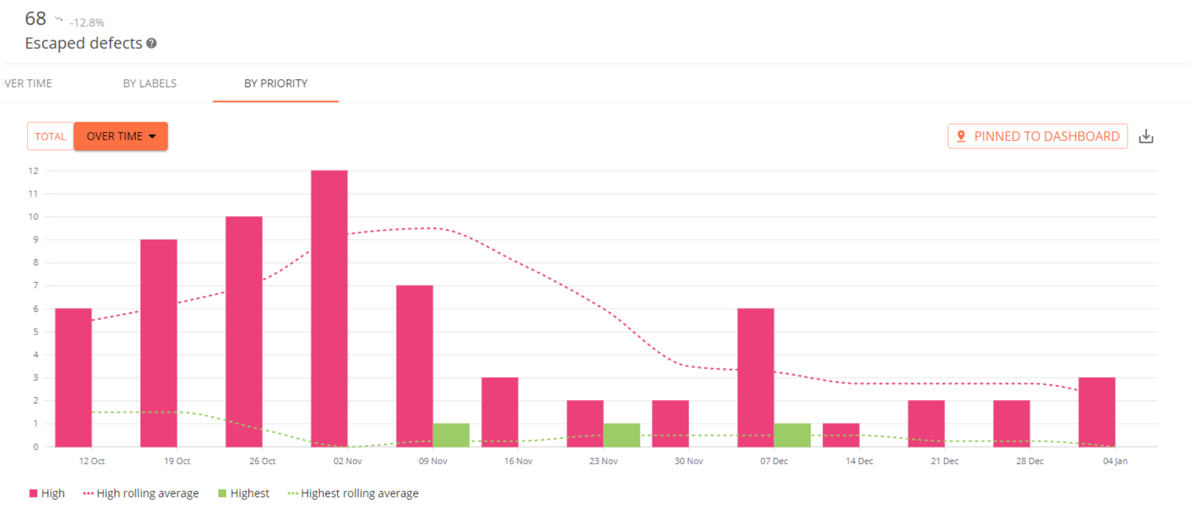
There may be concerns of teams 'gaming' the metric with story point inflation, but as with all metrics, they should be viewed in context by experienced folks who know the teams well. And if this is the case, they can stil give an excellent view of how the delivery organisation is progressing over time.

Figure 4 – Example Plandek Delivered Story Points metric view



And finally **Escaped Defects** is a simple but effective measure of overall software delivery quality. It can be tracked in a number of ways, but most involve tracking defects by criticality/priority as per the example below.

Figure 5 – Example Plandek Escaped Defects metric view



When these four simple Agile delivery metrics are viewed together, the early stage Agile DevOps practitioner can get a good balanced view of how their Agile DevOps maturity is progressing.

The metrics can be tracked over time, making sure that an improvement in one metric (e.g. Cycle Time) does not lead to a detrimental effect on another metric (e.g. Escaped Defects).

In addition, the relationship between Cycle Time and Deployment Frequency can be closely watched. Very often teams are able to reduce their Cycle Time, but this does not translate into quicker value delivery, due to bottlenecks in the integration and deployment process.

Metrics for intermediate Agile DevOps practitioners

As organisations progress in their Agile transformation, they may start to consider a more complete set of Agile delivery and DevOps metrics.

Our suggestion would be that you may start to implement a hierarchy of cascading metrics. These can be separated into:

1. 'North Star' leadership metrics - to be adopted by the leadership team to set the overall direction for the delivery organisation
2. Team and competence metrics – to be adopted by key players within the delivery organisatoin such as Team Leads, Delivery Managers, Product Managers, Engineering Managers and DevOps Managers. These are relevant to the areas in question and help drive improvement in the aggregate 'North Star' metrics adopted by the technology leadership team.

We would suggest that the early-stage metrics (discussed above) provide a sensible start point of 'North Star' metrics for the 'intermediate' Agile DevOps practitioner as they underpin the fundamentals of increased agility and are relatively easy to understand and act upon.

Figure 6 – Intermediate stage Agile DevOps Maturity metrics

Intermediate stage DevOps Maturity Metrics – 'North Star' technology leadership metrics	**Intermediate stage DevOps Maturity Metrics – Team and Competence Metrics**
1. Lead Time 2. Cycle Time	Flow Efficiency Mean Time to Resolve Pull Requests First TIme Pass Rate
3. Deployment Frequency	Number of Builds Build Failure Rate

	Deployment Cycle Time
4. Throughput (Delivered Story/Value Points)	Stories Delivered by Epic Delivered Value Points.
5. Escaped Defects	P1 Resolution Time Unresolved P1/P2 Bugs Tickets without a Pull Request

The Team and Competence metrics have been selected as powerful determinant metrics that directly drive the 'North Star' leadership metrics that track your core progress towards Agile DevOps maturity.

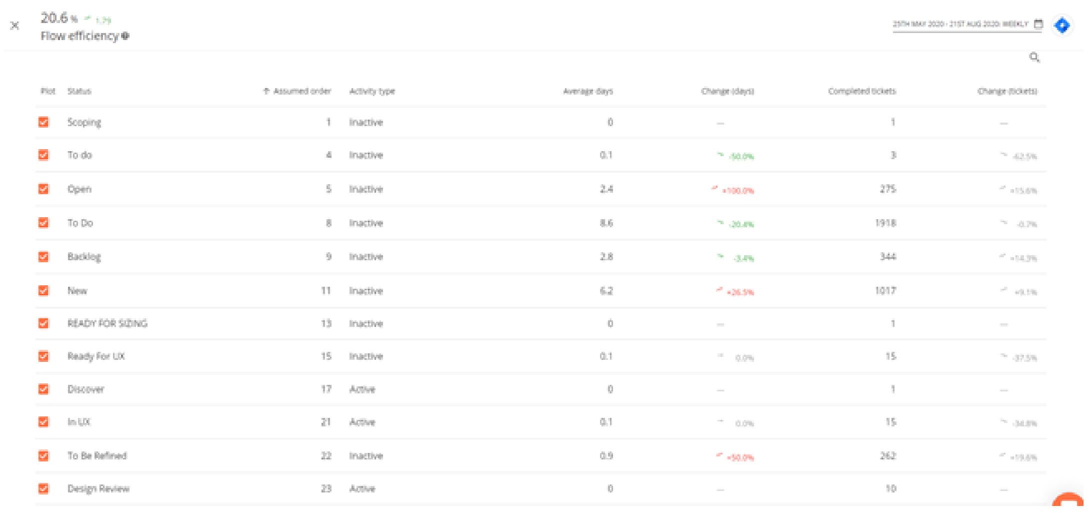
We have selected three metrics that in our experience most directly increase velocity (and hence reduce Lead and Cycle Time). These are:

1. **Flow Efficiency** (which looks at the proportion of time tickets spend in an 'active' versus 'inactive' status)
2. **Mean Time to Resolve Pull Requests** (hrs)
3. **First Time Pass Rate** (%).

Typically these metrics are adopted by each scrum team and related Scrum Masters and Delivery Managers, so that they are tracked and analysed in daily stand-ups, sprint retrospectives and management review meetings.

The Flow Efficiency analysis (see Figure 2 below), enables Team Leads to isolate and analyse each 'inactive' status in the workflow and consider if there is scope to reduce or eliminate it. The analysis shows the relative size of each 'inactive' status opportunity in terms of time spent in the inactive state and volume of tickets affected.

Figure 7 - Example Flow Efficiency metric within Plandek dashboard



20.6 % 1.79
Flow efficiency

25TH MAY 2020 - 21ST AUG 2020: WEEKLY

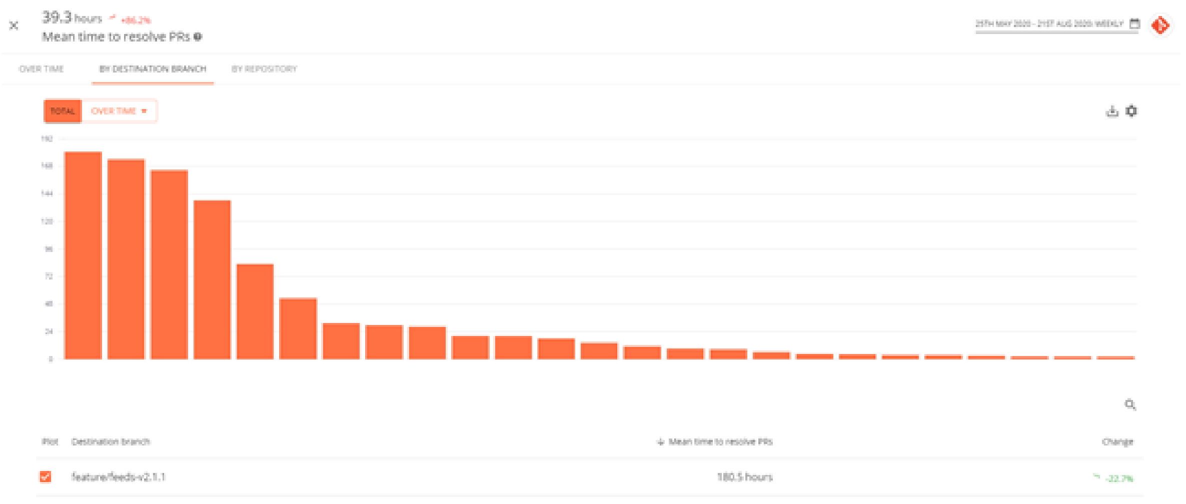
Plot	Status	Assumed order	Activity type	Average days	Change (days)	Completed tickets	Change (tickets)
✓	Scoping	1	Inactive	0	—	1	—
✓	To do	4	Inactive	0.1	-50.0%	3	-62.5%
✓	Open	5	Inactive	2.4	+100.0%	275	+15.6%
✓	To Do	8	Inactive	8.6	-20.4%	1918	-0.7%
✓	Backlog	9	Inactive	2.8	-3.4%	344	+14.3%
✓	New	11	Inactive	6.2	+26.5%	1017	+9.1%
✓	READY FOR SIZING	13	Inactive	0	—	1	—
✓	Ready For UX	15	Inactive	0.1	0.0%	15	-37.5%
✓	Discover	17	Active	0	—	1	—
✓	In UX	21	Active	0.1	0.0%	15	-34.8%
✓	To Be Refined	22	Inactive	0.9	+50.0%	262	+19.6%
✓	Design Review	23	Active	0	—	10	—

Typical opportunities to remove inactive bottlenecks included time spent with tickets awaiting definition (e.g. Sizing) and tickets awaiting QA. Where waits for QA were considered excessive, Delivery Managers reconsidered QA resource allocation by team.

Mean Time to Resolve Pull Requests (MTRPR) was also found to be a key bottleneck and hence potential area to save time and reduce overall Cycle Time. Very significant variations in time to resolve PRs were seen between teams and individuals, with waits of over 50 hours not uncommon.

Plandek enables drill-down to understand variances by code repository and destination branch (see Figure 8 below). This enabled quick identification of the biggest bottlenecks and targeted intervention, with the result that MTRPR was reduced dramatically by an average of c50%. This has a very significant impact on overall Cycle Time.

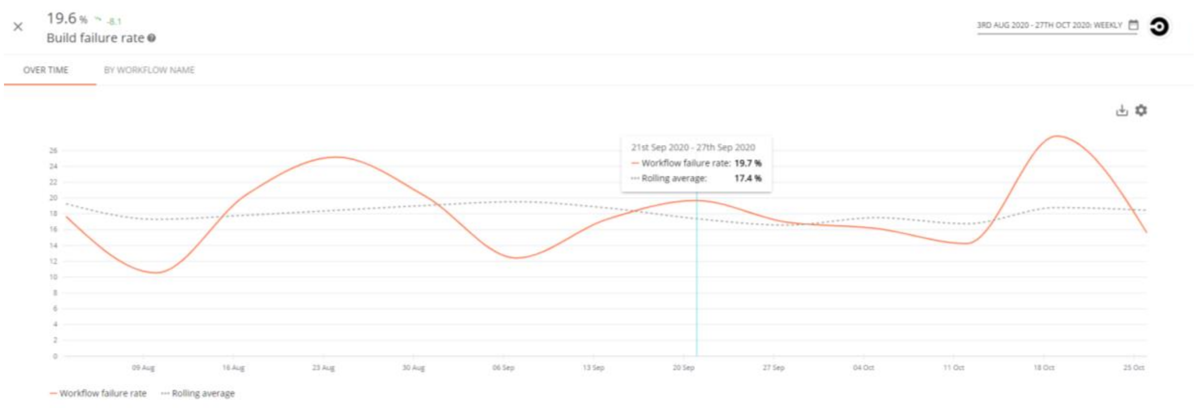
Figure 8 – Example Mean Time to Resolve Pull Request metric within Plandek dashboard



Plot	Destination branch	Mean time to resolve PRs	Change
✓	feature/feeds-v2.1.1	180.5 hours	-22.7%

In keeping with the ‘North Star’ metric of increasing **Deployment Frequency,** DevOps practitioners can track a range of metrics including: **Number of Builds, Build Failure Rate** and **Deployment Cycle Time**. All three are simple metrics which directly impact Deployment Frequency.

Figure 9 - Example Build Failure Rate metric

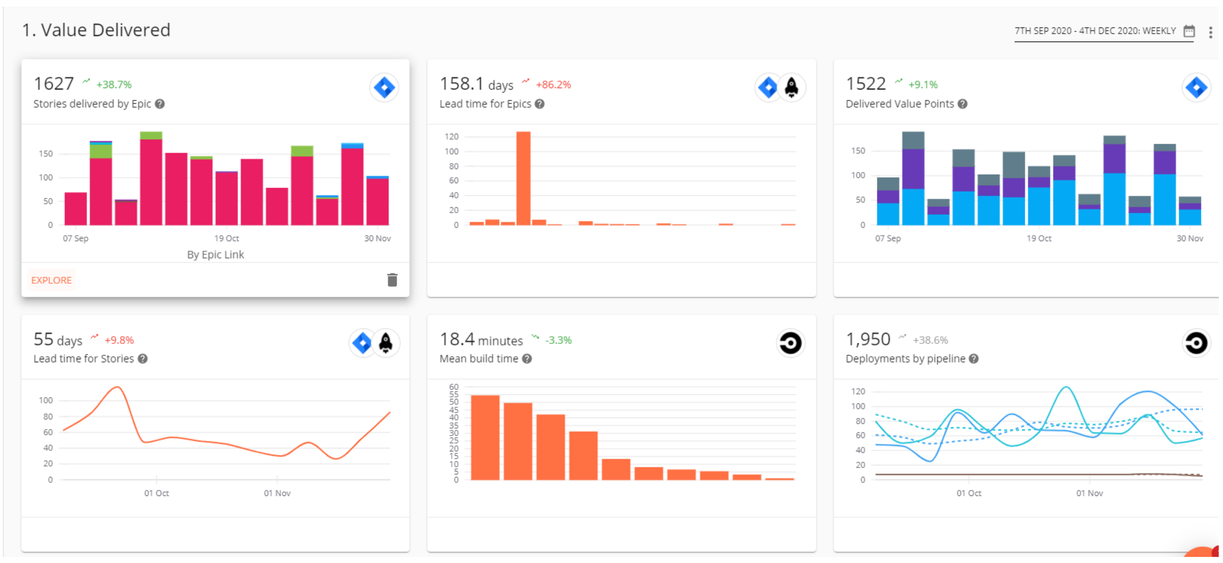


Our experience shows that typically you can expect to **increase deployments per day** (per pipeline) by 15% through a better understanding of the root-cause of Build Failures and Deployment Cycle Time using Plandek.

Delivery Team Leads and Managers can adopt a range of determinant metrics that help track and drive throughput and the delivery of value.

These included **Stories Delivered by Epic, Lead Time for Stories and Epic,** and **Delivered Value Points** (see Figure 10 below).

Figure 10 - Example value delivered metrics

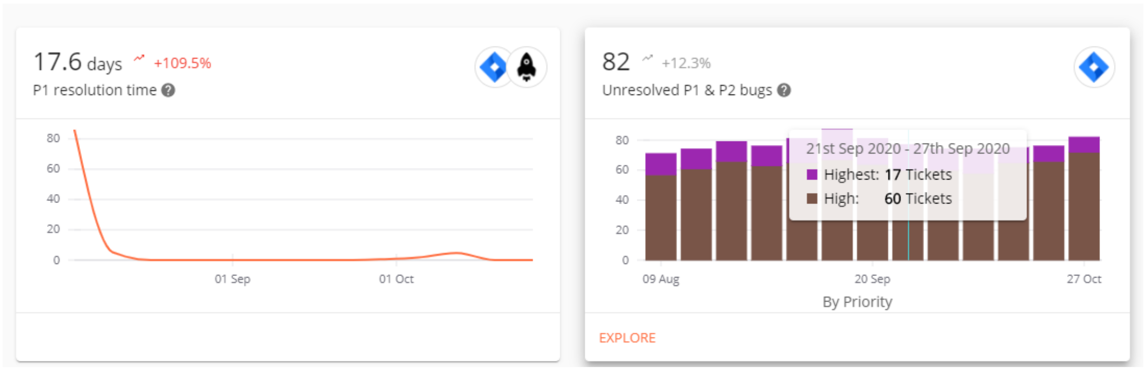


And finally quality should also be a consistent focus - both the security and quality of the delivery process itself and the quality of the software delivered.

We have selected a few simple Team and Competence metrics that directly impact defect rate - to reduce time spent fixing P1 (high priority) bugs, to improve the customer experience and reduce time diverted from feature development.

Plandek's customisable dashboards enabled each team to focus on their own P1 resolution time and to better manage the backlog of Unresolved P1 and P2 bugs and time to resolve key hot fixes.

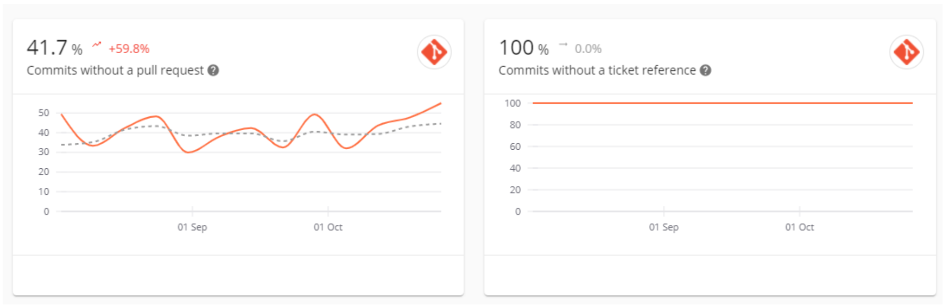
Figure 11 - Example quality metrics: P1 Resolution Time and Unresolved Bugs



We recommend that organisations at an intermediae level of Agile maturity start to adopt a broad view of 'quality' to include both the software output and the quality and security of the delivery process itself. As such teams may also track and manage **Commits without a Pull Request** and **Commits without a Ticket Reference.** See Figure 12.

The former ensures that all code is peer-reviewed before being committed (an important security requirement) – and the latter ensures the clear linkage between committed code and Jira tickets, for security compliance.

Figure 12 - Example delivery process quality metrics



Metrics for advanced Agile DevOps practitioners

As organisations reach an advanced level of Agile capability, they will no doubt have their own favoured metrics and analytics. With that in mind, we have selected our favourite metrics for mature agile delivery organisations.

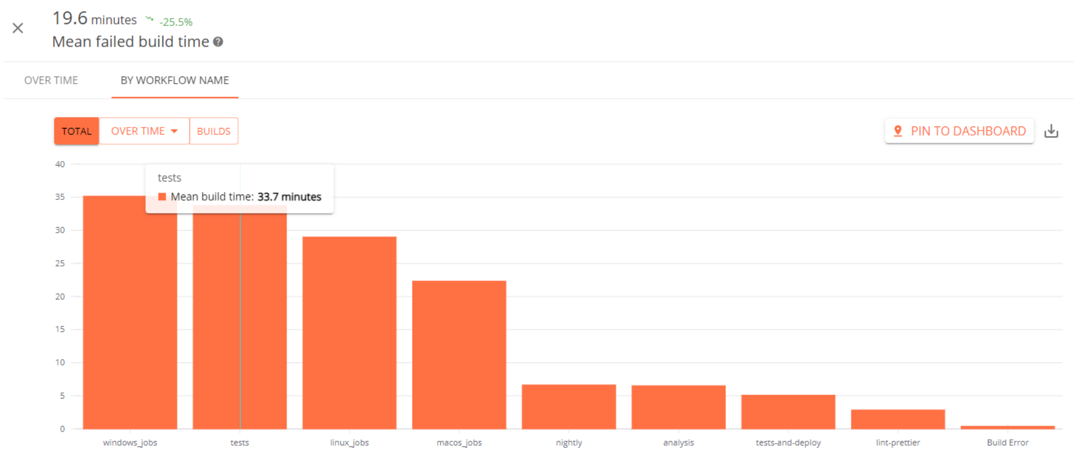
Again, our suggestion is that you maintain a hierarchy of cascading metrics, separated into North Star' leadership metrics to set the overall direction for the delivery organisation; and Team and Competence metrics – to be adopted by key players within the delivery organisatoin such as Team Leads, Delivery Managers, Product Managers, Engineering Managers and DevOps Managers.

As an example, mature agile delivery organisations are often very focused on refining the CI/CD process to increase deployment frequency and hence regular delivery of value to the organisation. As such, additional metrics such as *Failed Build Recovery Time* and *Mean Time for Failed Builds* become popular as teams try to track and reduce the impact of build failures. *Flakiest Files* is another specialist DevOps metric developed by Plandek which enables DevOps Managers to identify fragile source code files in their codebase which can be targeted for refactoring in order to reduce failed builds.

Figure 13 – Advanced stage Agile DevOps Maturity metrics

Advanced stage DevOps Maturity Metrics – 'North Star' technology leadership metrics	**Advanced stage DevOps Maturity Metrics – Team and Competence Metrics**
Lead Time Cycle Time	Flow Efficiency Mean Time to Resolve Pull Requests First Time Pass Rate
Deployment Frequency	Number of Builds Build Failure Rate Deployment Lead Time Mean time for failed builds Failed build recovery time Flakiest files
Throughput (Delivered Story/Value Points)	Stories Delivered by Epic Delivered Value Points.
Escaped Defects	P1 Resolution Time Unresolved P1/P2 Bugs Tickets without a Pull Request

Figure 14 – Example Mean Failed Build Time metric- showing analysis by workflow name


19.6 minutes -25.5%
Mean failed build time
OVER TIME
BY WORKFLOW NAME
TOTAL
OVER TIME
BUILDS
PIN TO DASHBOARD
tests
Mean build time: 33.7 minutes
40
35
30
25
20
15
10
5
0
windows_jobs
tests
linux_jobs
macos_jobs
nightly
analysis
tests-and-deploy
lint-prettier
Build Error