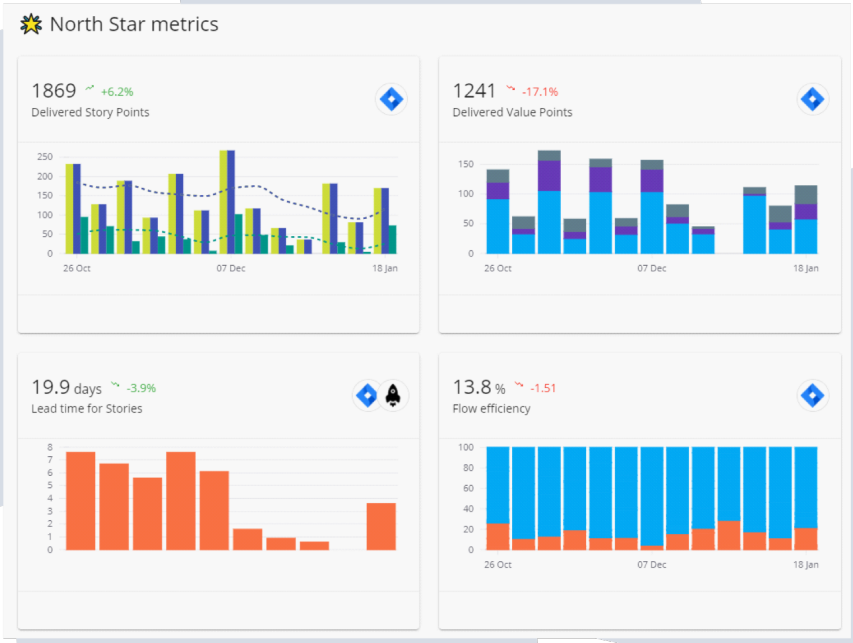
Plandek

The Ultimate Guide to Software Delivery and Engineering Metrics



Authors:

- **Charlie Ponsonby, Co-CEO Plandek**
- **Will Lytle, Customer Success Director, Plandek**

Contents

Introduction

As DevOps Value Stream Management (VSM) goes mainstream, organisations large and small increasingly recognise the need to apply data analytics to more effectively manage the end-to-end software delivery process – in order to deliver quality software faster and more predictably.

Plandek is a global leader in end-to-end software delivery metrics and analytics and we work with clients of all sizes and levels of agile DevOps maturity.

This 'ultimate guide' is designed as a practical resource to help you select the metrics that you might want to track (relevant to your use case); the tools you need to collect the data; and how you might set targets, embed the metrics across your organisation and drive real behaviour change, in order to improve your delivery effectiveness.

The guide draws on our experience from multiple clients who use Plandek's end-to-end software delivery analytics to improve their delivery capability and outcomes. For more details of Plandek please visit www.plandek.com.

METRICS

1. The rise and rise of end-to-end software delivery metrics

Delivery metrics were well used in the pre-Agile 'waterfall' era. But that changed with the arrival of Agile software delivery twenty years ago.

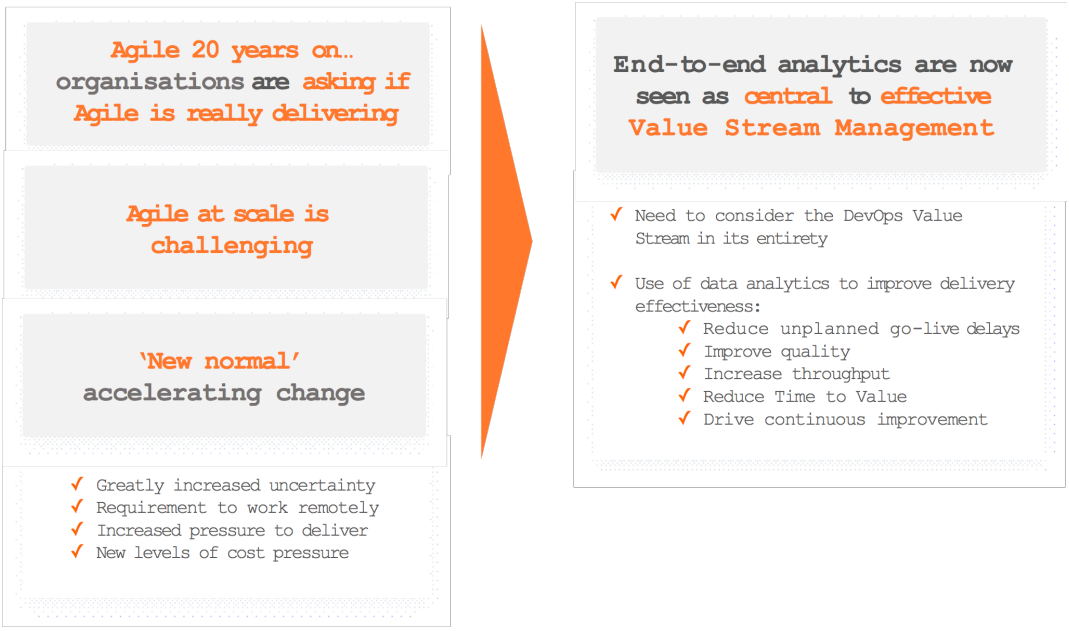
The Agile Manifesto sets out a better way to deliver software. It is based on some core 'democratic' principles that empower individuals and teams to be self-determining and to define their own work schedule and processes. As such it has (on-balance) been very successful and has been adopted by over 80% of enterprises globally in some form.

A core element of this culture of self-determination has been a healthy scepticism of top-down metrics and analytics.

However, the Agile approach is now 20 years old and well past the 'honeymoon period'. As a result, larger enterprises particularly (who face the challenge of implementing Agile at scale) – now recognise that data analytics can play a crucial role in effective Agile software delivery. Hence the current explosion in interest in software delivery metrics.

As shown in Figure 1 below, the pressures of the 'new normal' world have only accelerated the recognition of the importance of analytics to improve delivery effectiveness.

Figure 1 – The role of end-to-end delivery metrics in effective DevOps Value Stream Management

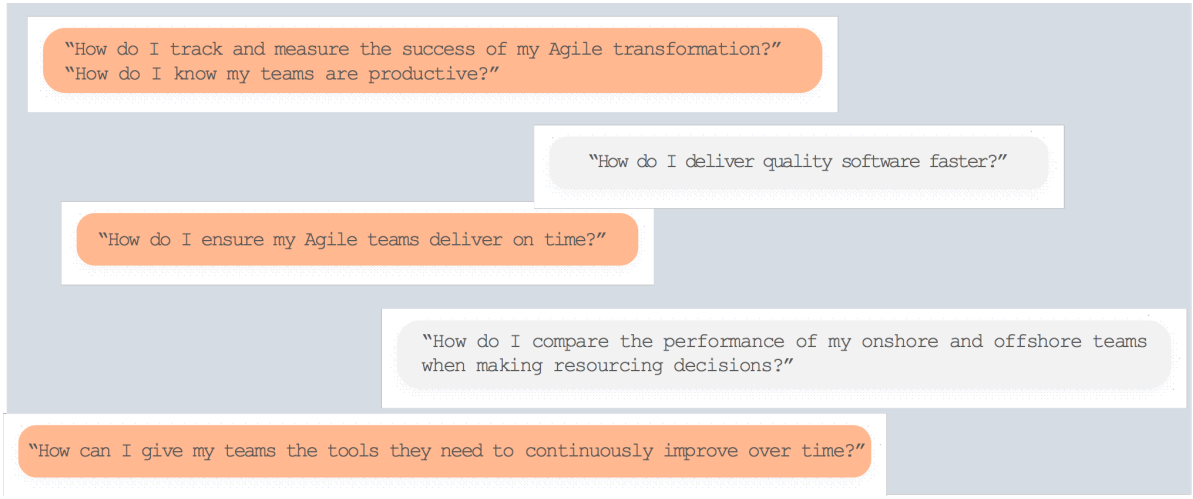


It is widely recognised now that software delivery metrics and analytics can help improve delivery effectiveness in multiple ways:

- remaining Agile and delivering software more dependably (predictably), in keeping with the timing requirements of customers
- improving the quality and security of both the software itself and the process of software delivery
- increasing velocity and increase the throughput of value delivered (reducing Time to Value)
- and empowering teams to self-improve over time.

As such, delivery metrics can play a critical role in many widely experienced challenges (use cases) in delivery organisations as shown in Figure 2 below.

Figure 2 – some typical challenges that end-to-end delivery metrics can help solve



2. End-to-end metrics in effective DevOps Value Stream Management

This recognition of the importance of end-to-end metrics in effective Agile software delivery has coincided with the growth of the concept of DevOps Value Stream Management. (See Gartner Market Guide Sept 2020: 'DevOps Value Stream management Platforms').

Value Stream Management (VSM) encourages technology teams to view Agile delivery as an-end-to-end value delivery process that requires careful orchestration and management. The view is that end-to-end metrics and analytics are critical to effective value stream management – to measure how effectively value is actually delivered.

> "Organisations lack end-to-end visibility into product delivery and struggle to improve their flow of value. I&O leaders...must implement a DevOps value stream management platform and analyze value stream metrics to optimize the overall health of product delivery."
> Source Gartner DevOps Value Stream Management Platforms Market Guide Sept 2020

3. Key watch-outs when applying software delivery metrics

3.1. The role of metrics in the 'philosophy' of Agile software delivery

As mentioned above, the Agile Manifesto is based on some core democratic principles that empower individuals and teams to be self-determining and to define their own work schedule and processes.

As such, a core element of this Agile culture has been a healthy scepticism of top-down metrics and analytics. Many Agile practitioners have tended to view metrics as potentially flawed in two key ways:

- they are somehow contrary to the 'spirit' of self-determination and the individual
- they are often unrepresentative, inaccurate and too easily gamed.

Hence despite the current explosion in interest in software delivery metrics, scepticism to metrics is often still prevalent in parts of organisations. Hence embedding metrics in Agile delivery organisations can be challenging.

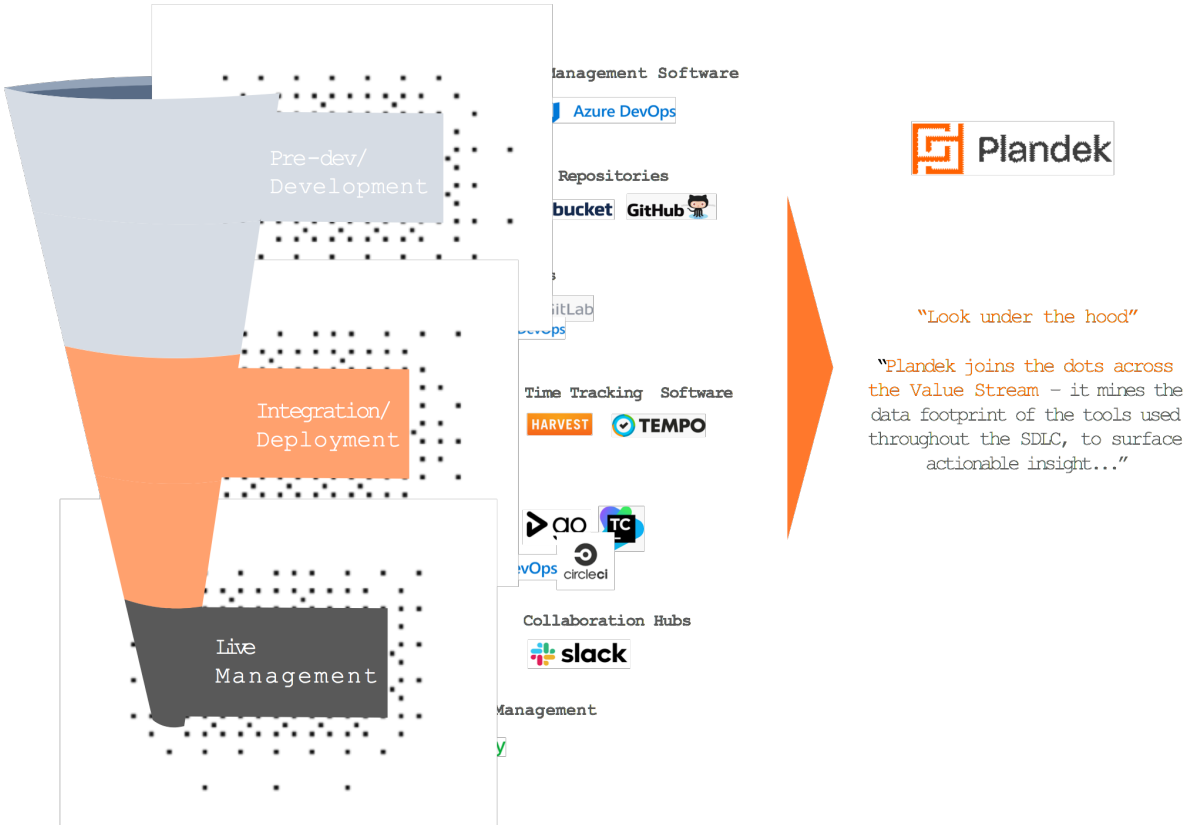
To succeed, it needs:

1. Strong technology leadership and sponsorship of a data-led approach to software delivery
2. The ability to easily surface meaningful and accurate metrics in near real-time which teams and individuals understand and trust
3. A framework and methodology to embed metrics across the delivery organisation and shift behaviour so that teams can set their own targets and self-improve around the metrics in question.

3.2. Data sources for effective software delivery and analytics

Software delivery is a complex and inter-related process. Useful analytics requires an end-to-end view of the inter-related processes across Pre-Development, Development, Integration, Deployment and Live Management. As an example, Figure 3 below shows the core Plandek data integrations.

Figure 3 – Multiple systems integrations required for an end-to-end view of the software delivery process



Collating, flattening and analysing data from these disparate sources is a complex task. It can be done manually or by applying a generic BI tool like Tableau, but it is resource intensive and prone to failures/errors as the underlying systems constantly change.

As such, a specialist end-to-end software delivery analytics tool like Plandek is very often the only viable solution.

3.3. Data security

Data security is critical in software delivery analytics. It requires access to data-sensitive and critical systems (including for example proprietary project information held within workflow management tools and source code held in code repos). Hence information security is always a key priority.

Most delivery analytics BI solutions are cloud-based, so much care is needed in selecting a data-secure solution. Plandek is used by infosec sensitive organisations as it addresses infosec in four ways:

1. It is secure architected in the European Google cloud

2. It cleans the base data to only analyse non-sensitive meta-data (e.g. it removes labels from Jira tickets)
3. It encrypts data before any data transfer
4. It offers an on-premise data gatherer solution, so that all sensitive data is held on-premises and only summary data presentation is undertaken in the cloud.

We recommend that you take account of all four of these considerations when selecting a software delivery analytics solution.

3.4 Metrics and analytics configurability

The 'devil really is in the detail' with software delivery analytics for two key reasons:

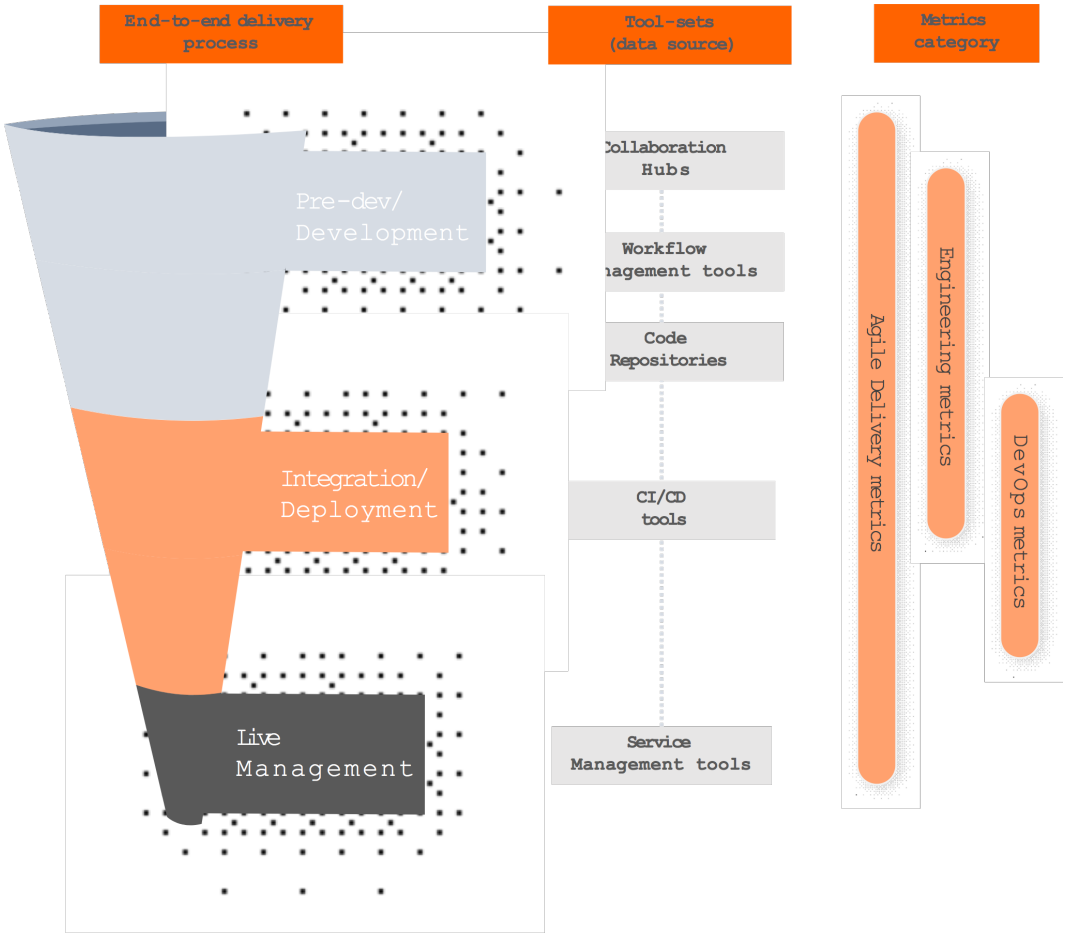
- First, the software delivery process is extremely complex (especially at scale) and may involve: separate system stacks; multiple systems and system instances; many different workflows and related operational complexity. As such, metrics are meaningless unless they can be very carefully configured to take into account of the context in which they are applied. Very often software delivery analytics fail as the metrics look plausible, but when scrutinised by the teams involved, they are discarded as they do not accurately reflect the situation 'on the ground'.
- Second, (and related to the first point) 'ownership and trust' are critical in any metrics roll-out there may be a scepticism among some members of the engineering team as to the suitability of metrics. As such ownership and trust in metrics will not be achieved across all teams unless the metrics very accurately reflect the idiosyncrasies of each team's particular situation/workflow. Indeed, if users start to distrust the metrics, any hope of adoption is doomed to failure.

It is very important therefore to check the integrity of the metrics surfaced before attempting roll-out and adoption. The BI tool needs to be flexible enough to ensure that metrics can be configured to accurately represent the truth and gain the trust of users at the team level. This can be done during a technical proof of concept, pre roll-out.

4. Delivery and Agile Metrics

Figure 4 below shows a summary of the end-to-end software delivery process, the core toolsets that underpin that process; and the key categories of metrics that are derived from the various data sources.

Figure 4 – High level view of end-to-end software delivery metrics by data source



The first category that we will consider is Delivery and Agile metrics which track the effectiveness of the end-to-end delivery process and therefore draw data from all the toolsets that sit across that process (see Figure 4 above).

We have divided delivery and agile metrics into four categories:

1. Value delivered
2. Delivery efficiency
3. Dependability (covered under the Sprint Metrics section)
4. Backlog health
5. Delivery quality.

4.1 Value Delivered Metrics

Figure 5 below shows our choice of 'value delivered' metrics. Delivery of value is always tricky to measure and many organisations end-up using the proxy measure of story points delivered (throughput), though some organisations may use 'value points' as an alternative to story points. Typical metrics include therefore:

Stories Delivered by Epic

Core metric that simply tracks stories delivered by Epic. As an Epic is designed to reflect a (valuable) increment of work that is meaningful to the organisation. This metric is therefore a decent proxy of value delivered by relevant increment of work (epic). You

may also want to consider *Stories Delivered by Sprint* and *Stories Delivered by Benefit* (if defined as a custom field)

Delivered Value Points

For those organisations using value points, this is clearly a critical metric and may also be viewed by Epic or Sprint for example

Clearly value is not delivered until software is actually deployed to live. Hence value delivered metrics should also measure total Lead Time (from the time a ticket leaves the backlog until the time deployed to live), rather than Cycle Time which typically just looks at the time a ticket spends in development (pre integration test and deployment).

Lead Time for Epics, Lead Time for Stories

Time is a critical factor in software delivery, so being able to understand Lead/Cycle Time and potential bottlenecks makes this a must-have. Lead Time is calculated by looking at completed tickets over time and adding up the elapsed time in calendar days (including weekends) of all statuses of those tickets. You then divide the total time by the number of completed tickets to derive the Lead Time.

As the Integration-test-deployment phase of the delivery process is a key potential bottleneck in value delivery, two other metrics we tend to include under the Value Delivered umbrella are *Mean Build Time* and *Deployments by Pipeline*

Mean Build Time

This is a helpful metric to identify slow build processes which affect the ability of the team to deliver software. A steadily increasing mean workflow time will want to be addressed and will drive longer cycle times. You can filter this metric by status to help keep an eye on slow builds which ultimately end in failure.

Deployments by Pipeline

As the name suggests this is an important measure of your ability to rapid deploy increments of software to live. It becomes more powerful when filters and breakdowns are applied by project or workflow name for example.

Figure 5 – Plandek screenshot showing example Value Delivered dashboard

4.2 Delivery Efficiency Metrics

Measuring throughput (or value delivered) is a very good start point, but it is clearly also very important to understand how efficiently you are delivering that value, in order to understand how you might increase velocity and deliver value more often with your existing resource.

Figure 6 below summarises some key Delivery Efficiency metrics.

Speeding Transitions Rate

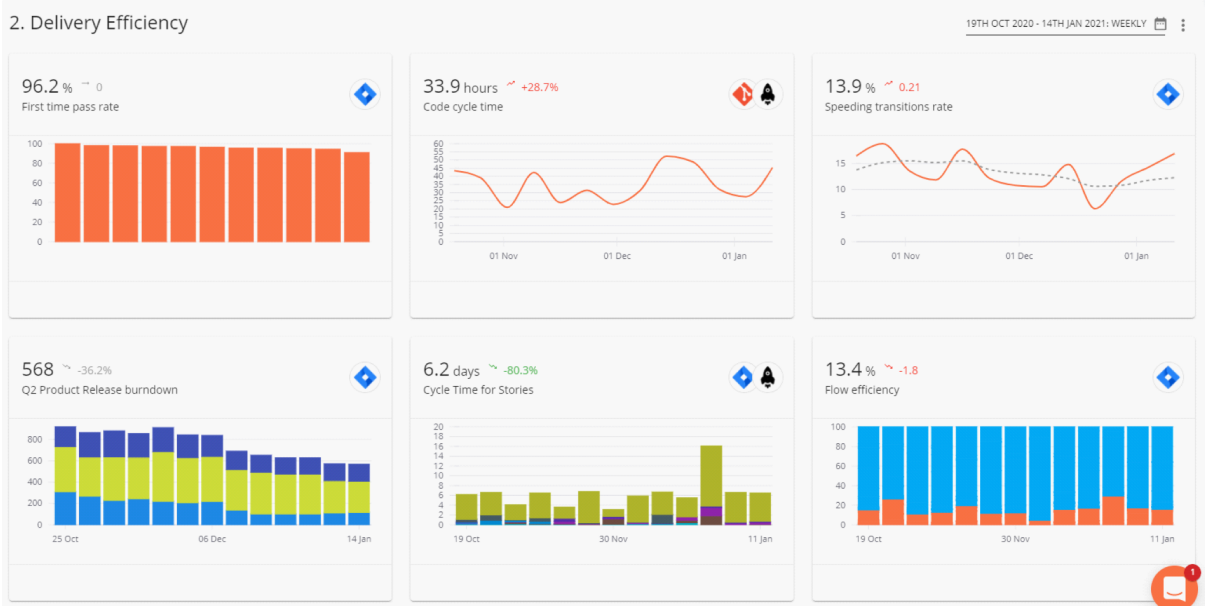
We have put this metric first as it may be a critical start-point for many delivery teams. If teams are not accurately using their workflow management tool (e.g. Jira), then visibility of the delivery process declines very rapidly. As such, it is very difficult to make meaningful efficiency improvements. This metric therefore tracks how accurately teams are using ticket statuses in Jira. Engineers may forget to update ticket status in real time and 'shepherd' tickets through multiple statuses at the end of a sprint to 'tidy up'. As such, you have no idea of the real duration of each status and when it really occurred. The Speeding transitions rate shows the proportion of tickets (by individual and team) that have been 'sped through the process' rather than updated in real time. This is therefore a critical place to start – to encourage teams to use Jira correctly so that the delivery process can then be analysed and improved.

Flow Efficiency

Flow efficiency is a critical measure of 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 very 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).

Figure 6 – Plandek screenshot showing example Delivery Efficiency dashboard



First Time Pass Rate

First Time Pass Rate is one of our favourite metrics as it is a very good proxy metric for overall delivery *process health* (rather than a reflection of the capabilities of the individual engineer). The metric refers to tickets successfully passing through QA without returning for additional work. A high FTPR reflects teams that are working well together, with well-defined requirements, a carefully prepared ticket backlog and highly engaged engineers – so that tickets have the highest probability of passing first time. Conversely, a low FTPR greatly reduces velocity and drains morale.

Cycle Time for Stories and *Code Cycle Time*

These metrics examine the efficiency of two key elements of the end-to-end delivery process. *Cycle Time for Stories* looks at the development time elapsed and time spent in each status within the development process, in order to identify bottlenecks. *Code Cycle Time* 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 (time to review, time to approve, time to merge/close, time to deploy).

4.3.Backlog Health

The software delivery process relies heavily on a prepared backlog of well-defined tickets waiting to be worked on. Clearly, if this backlog of tickets is too small or poorly defined, the effectiveness of the whole delivery process is put at risk. Hence Backlog Health metrics are another critical category of Agile Delivery Metrics.

Story Points Ready for dev

As the name suggests, this is the core Backlog Health metric. It shows the available backlog of defined tickets expressed in story points and should be considered relative to the velocity of teams to ensure that there is always 1-2 sprints of backlog 'cushion'. This ensures that teams always have well defined tickets to work on and never wait for tickets to be defined and/or are forced to work on poorly defined tickets.

Time to Design Stories

Tracks the time taken to design and prepare tickets. This is an important consideration when analysing the time it is likely to take to recover if there is insufficient backlog.

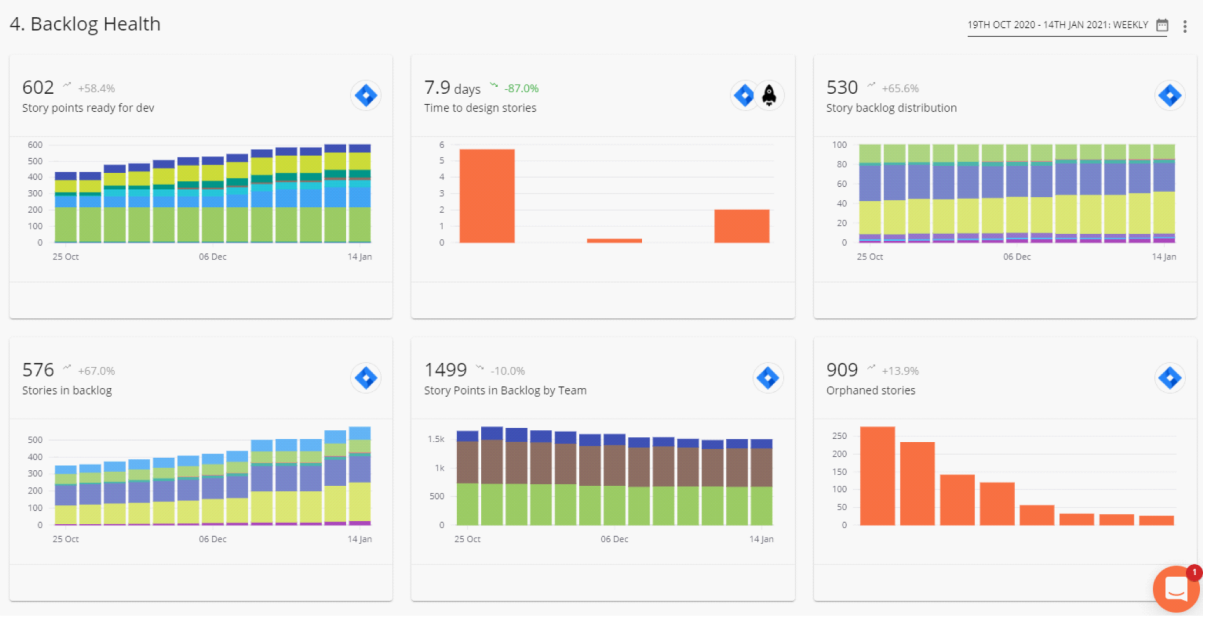
Stories in Backlog and *Story Points in Backlog by Team*

These are two further metrics to better understand the size and allocation of available backlog by team.

Story Backlog Distribution

This is a very useful metric to manage the backlog management process. It shows tickets in backlog by status (e.g. 'in design', 'in copy', 'to be refined') so that you can see where effort is required to maintain your backlog health and hence overall delivery health.

Figure 7 – Plandek screenshot showing example Backlog Health dashboard



Orphaned Stories

All stories should be linked to a 'parent' (e.g. an epic), rather than unassigned or 'orphaned'. This process should take place at the outset when a story is prepared and placed in the backlog, hence this metric is another useful measure of backlog health.

4.4. Delivery Quality

There are very many Delivery Quality metrics that can be applied. We have selected three simple quality metrics.

Escaped Defects

This is a good summary quality metric which tracks the number of Escaped Defects (bugs) identified over time. The defects can be tracked by severity and remains a fundamental measure of the quality of software output. It can be expressed as Defect Density for a more representative view of quality, which is often defined as the number of defects per 1,000 lines of code or function points.

Unresolved P1 and P2 Bugs

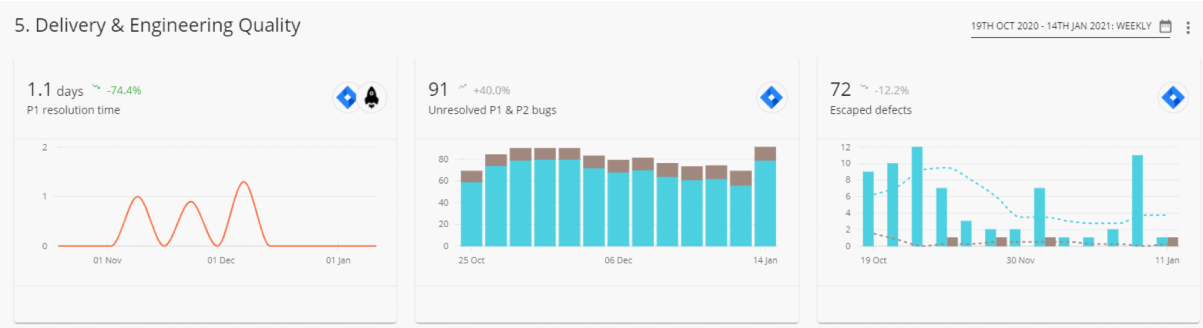
This metric is useful in tracking the impact of poor quality on the delivery process. An increase backlog of priority 1 and 2 bugs is not only bad for the end-user, but also bad for the delivery organisation, that will struggle to deliver new features with the bug backlog hanging over the teams.

P1 Resolution Time

This is related to the Unresolved Bugs metric as it allows managers to understand the impact of the backlog of unresolved bugs both for the end-user and the delivery team. A long P1 Resolution Time

suggest that customers will be negatively impacted by the bug for a prolonged period, whilst symptomatic of a delivery team/process that is unable to operate effectively.

Figure 8 – Plandek screenshot showing example Delivery Quality dashboard



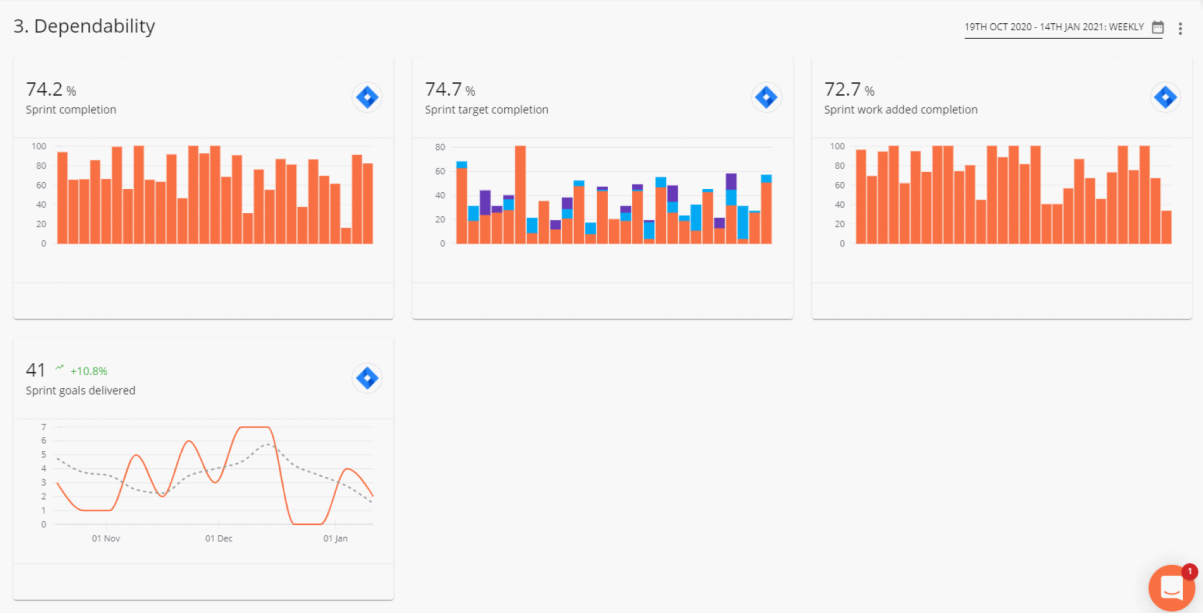
5. Sprint (Dependability) Metrics

Sprint or Dependability metrics (shown below in Figure 9) are a vital element of the overall Agile Delivery metrics 'balanced scorecard'.

One of Agile's biggest challenges and criticisms (especially in large organisations) is the inability for Agile teams to predict their output. All organisations will from time-to-time require software to be delivered within a certain timeframe in order to meet the needs of the stakeholder (e.g. due to a seasonal trading period, regulatory requirement or competitive pressure). In these instances, Agile teams have to provide visibility of timing of increments of output. And in order to be able to provide this visibility, the *dependability* of teams is absolutely vital.

To put it another way, if teams (in a scrum agile environment) cannot accurately deliver their own sprint goals (i.e. predict their output over a two week sprint), then it becomes nearly impossible to predict output across many teams and longer time periods (e.g. Programme Increments and Release Trains in a Scaled Agile context). As such, the team and the sprint are the basic building blocks for dependability of output. Teams that regularly hit their sprint goals can be relied on and hence broader commitments to stakeholders (at product or programme level) can be made with more confidence.

Figure 9 – Plandek screenshot showing example Dependability (Sprint Metrics) dashboard



Sprint Completion and *Sprint Target Completion*

These are the two most vital dependability metrics. They track scrum teams' ability to deliver their sprint goals (usually expressed in story points). *Sprint Completion* measures the percentage of story points actually completed at the end of the sprint time period, including story points that may have been added during the sprint. This is a valuable measure as it takes into account a key problem in sprint delivery – the fact that teams tend to add story points during the sprint period, due to poorly estimated tickets that get re-sized, or work added from another source. *Sprint Target Completion* measures excludes these added story points to measure what proportion of the story points planned at the sprint outset, where completed during the sprint timeframe.

Sprint Work Added Completion

As the name suggest, this metric looks at the proportion of work added during the sprint that is completed during the sprint.

Sprint Goals Delivered

Some organisations define clear goals for each sprint, which are deemed 'completed' or 'not completed' at the end of the sprint. There is a binary decision with no concept of 'half completed'. This can also therefore be a very good measure of teams' dependability.

6. Code Metrics

There are a myriad of potential code and code quality metrics. We focus here on code metrics that best reflect the *process* of writing code and therefore can improve the effectiveness of that process.

Commits Count

This metric considers the speed and agility of the engineering team as it tracks the volume and frequency of commits. It tracks the volume of code commits made to any branch based on the date of commit. As well as seeing the total number of commits over time, you can also analyse the frequency and size of commits by engineers and repositories.

Pull Request Count

Pull requests are at the heart of software development, and this metric allows you to gain key insight into your team's behaviour around this practice. Being able to cross reference this data with Jira/Azure enables you to understand how this behaviour is distributed across stories, tasks and bugs.

Commit hotspots

Many tools enable you to track where high levels of activity in your code base, but Plandek enables you to distinguish whether this activity is related to work on stories, bugs or other issue types within your workflow management tool. Now you can quickly identify where you're addressing technical debt or iterating new functionality quickly.

Ticket complexity

Many teams rely on anecdotal information about where levels of complexity lie in the code base, especially when building new functionality. With ticket complexity, you can quickly cross-reference critical repository activity with your workflow management tools to identify which stories, bugs, tech debt etc. are resulting in higher risks behaviours.

Two important metrics that track the quality of the engineering process are Commits without a Pull Request and Commits without a Ticket Reference. Both metrics are important from an infosec and process integrity point of view.

Commits without a Pull Request

This metric tracks the percentage of code commits that have occurred (tracked by team and individual), for which there is no assigned Pull Request – hence the code has been committed without peer review. Most organisations will view this as a major infosec breach, but it is a surprisingly common occurrence.

Commits without a Ticket Reference

This metric tracks code commits that have no assigned workflow management (e.g. Jira) ticket reference. This makes it impossible to trace issues in the code base back to the ticket in question. Root

cause analysis becomes much easier if effective tracking of ticket references is in place.

7. DevOps and Engineering Metrics

This category covers a broad range of potential metrics. We focus on those metrics we believe key to the end-to-end software delivery process and the health of the engineering capability underpinning that process.

The DevOps metrics that we list here reflect the core Agile objective of increasing the frequency of deployments by better managing the continuous integration and continuous deployment process.

As such the metrics chosen reflect the key objectives of:

- reducing build failure rate – a major source of friction in the process
- reducing time to build and time to recover from failed builds – another critical determinant of deployment efficiency; and
- streamlining the Pull Request process in order to optimise the time it takes to go from commit to deployment.

Number of Deployments and Deployment Frequency

Tracks the number of deployments to live and the frequency of those deployments. These a core Agile metric as the underlying objective of Agile (as stated in the Manifesto) is "the **early** and **continuous** delivery of valuable software". As such, this is the ultimate test of our ability to deliver software in an agile way.

Number of Builds and Build Frequency

Related to deployments is the number and frequency of builds (relating to the build steps leading up to a deployment such compile, generate code, package etc)

Mean Build Time and Mean Failed Build Time

Mean Build Time analyse the average time taken to execute a build which is very often a critical metric owing to lengthy build times affecting overall Lead Time. Of particular importance is *Mean Failed Build Time* with the old adage that if you are going to fail, it is better to fail fast. Identifying these builds enables teams resolve issues and to move more complex steps in the build process earlier in order to minimise the time that teams are down during builds.

Build Failure Rate and Mean Time to Recover from Failures

These metrics are often extremely helpful. Build Failure Rate looks at the percentage of failed builds and can be filtered by workflow. Failed Builds are a significant risk to delivery, both in slowing the process and creating additional work to respond to the incident (which is tracked in *Mean Time to Recover from Failures*).

Flakiest Files

Flakiest Files (which is only available in the Plandek dashboard) correlated 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 more effectively.

Figure 10 – Plandek screenshot showing an example DevOps metrics dashboard

8. Metrics for different levels of Agile DevOps Maturity

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 section 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.

8.1 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 measurable 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.

8.2 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 1 - Early-stage Agile DevOps Maturity metrics

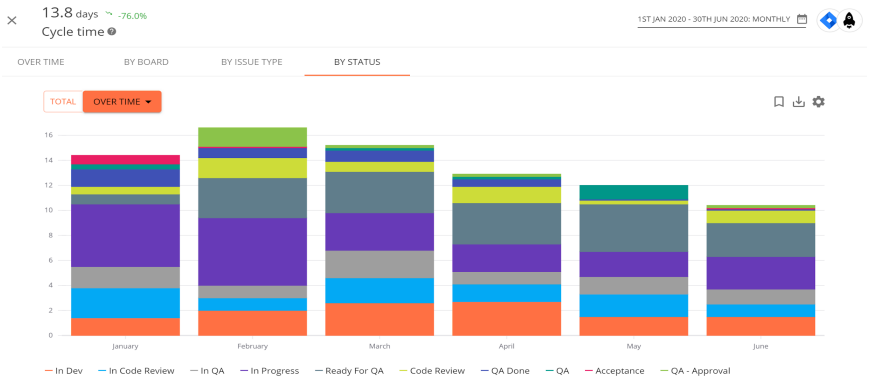
Early-stage DevOps Maturity Metrics	
1. Cycle Time	Early and continuous delivery
2. Deployment Frequency	

3. Throughput (Delivered Story Points or Value Points)	Delivery of value
4. Escaped defects	Quality of delivery

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.

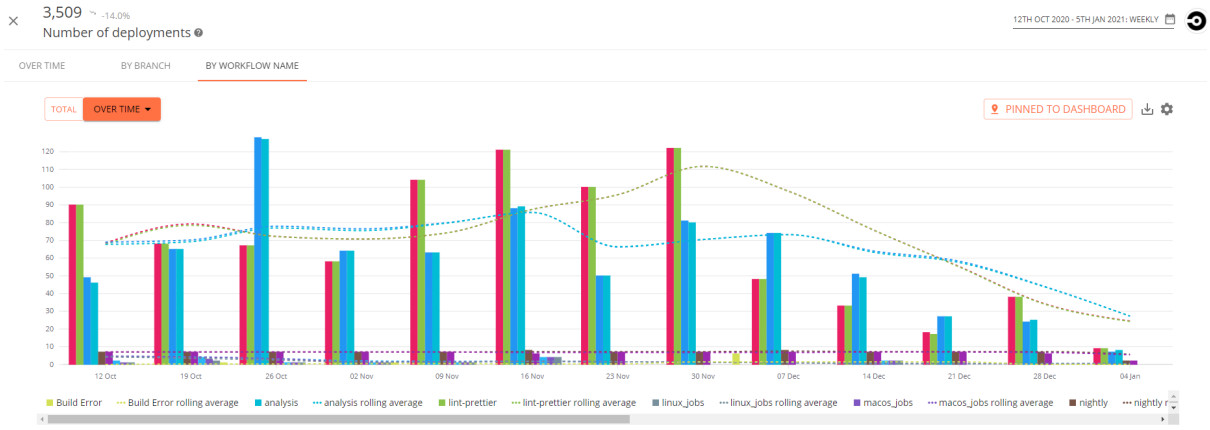
As per figure 12 below, the Cycle Time metric view allows teams to understand time spent in each ticket status within the development cycle. Plandek has flexible analytics capability and powerful filtering to allow analysis by Status, Issue Type, Epic (and any other standard or custom ticket field) all plotted over any time range required.

Figure 12 – Example Plandek Cycle Time metric view



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 13 – Example Plandek Deployment Frequency metric view



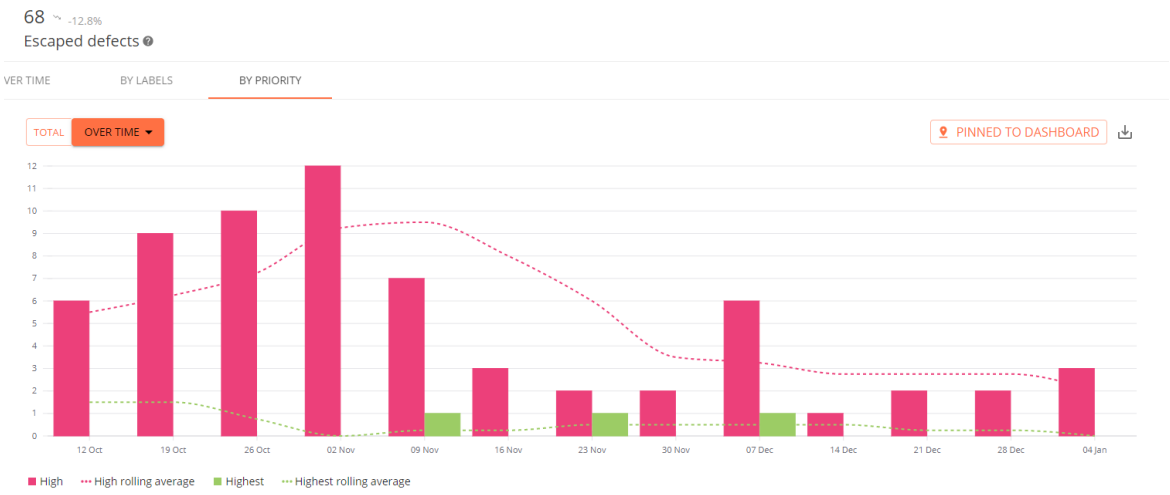
Delivered Story Points is often considered a problematic metric due to the potential inconsistencies in the calculation of story points and how much effort they represent. However, as a basic measure of output and how that is changing over time, it is a powerful metric around which to align.

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 still give an excellent view of how the delivery organisation is progressing over time.

Figure 14 – 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 15 – 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.

8.3 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 organisation 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 16 – 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 Lead 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.

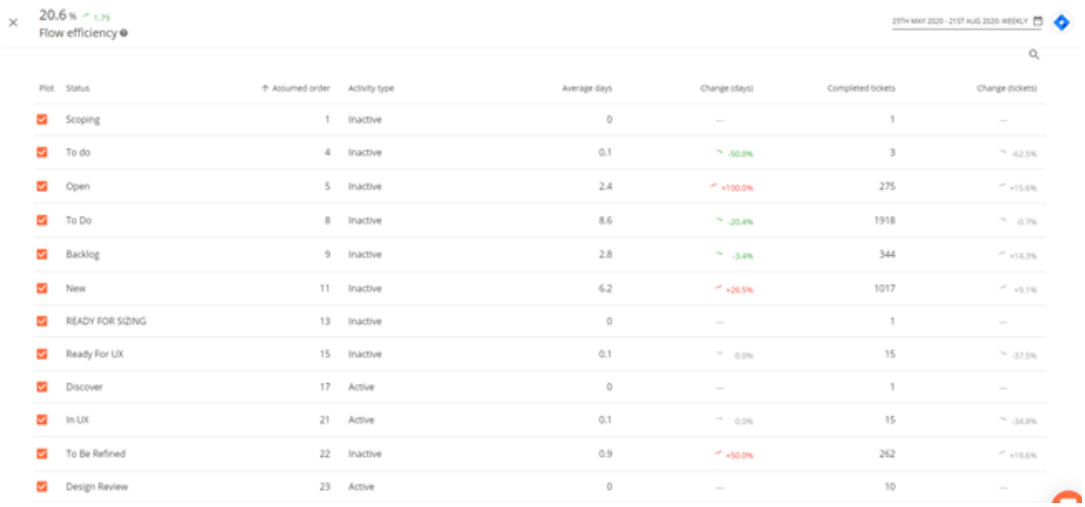
We have selected three metrics that in our experience most directly increase velocity (and hence reduce Lead and Cycle Time). These are:

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 17 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 17 - Example Flow Efficiency metric within Plandek dashboard

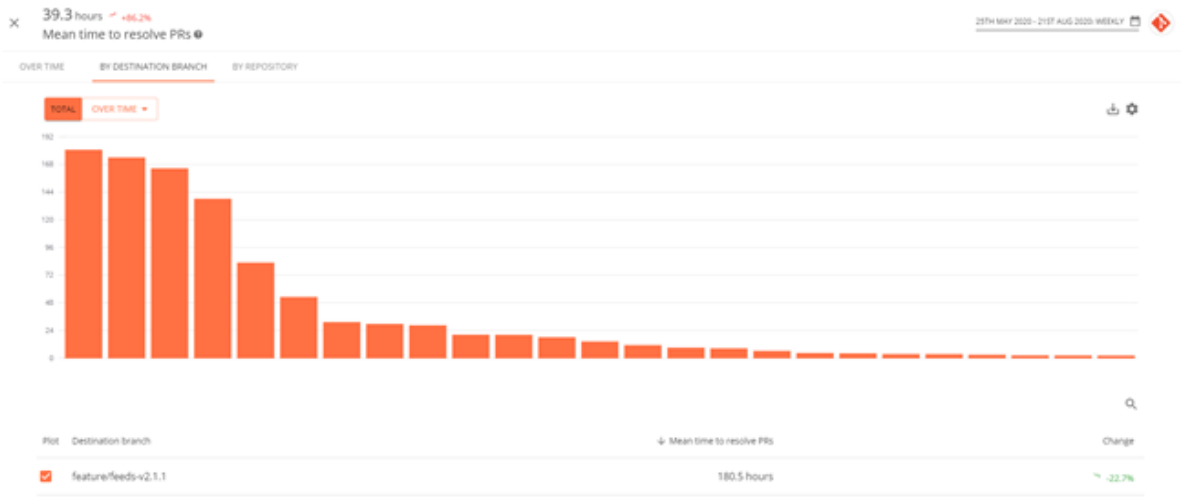


Typical opportunities to remove inactive bottlenecks included time spent with tickets awaiting definition (e.g. Sizing) and tickets awaiting QA. Where waits for QA are considered excessive, Delivery Managers can reconsider QA resource allocation by team.

Mean Time to Resolve Pull Requests (MTRPR) is also often 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 are seen between teams and individuals, with waits of over 50 hours not uncommon.

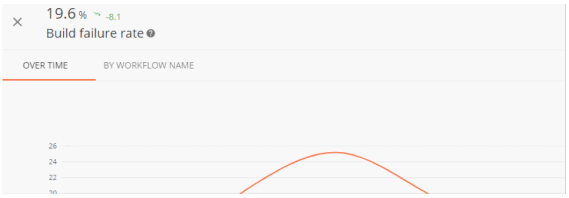
Plandek enables drill-down to understand variances by code repository and destination branch (see Figure 18 below). This enables quick identification of the biggest bottlenecks and targeted intervention, with the result that MTRPR can be reduced dramatically. This has a very significant impact on overall Cycle Time.

Figure 18 – Example Mean Time to Resolve Pull Request metric within Plandek dashboard



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 Lead Time*. All three are simple metrics which directly impact Deployment Frequency.

Figure 19 - 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.

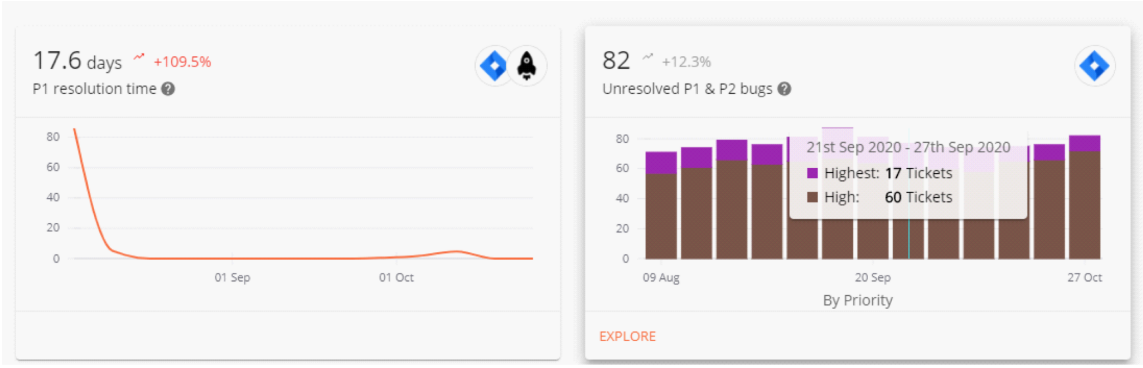
These included *Stories Delivered by Epic, Lead Time for Stories and Epic*, and *Delivered Value Points*.

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 20 - Example quality metrics: P1 Resolution Time and Unresolved Bugs



We recommend that organisations at an intermediate 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*.

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.

8.4 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 organisation 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 21 — 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

APPLYING METRICS TO DRIVE BUSINESS BENEFIT

9. Typical use cases for the application of end-to-end software delivery metrics & analytics

Figure 22 summarises the most common use case for the application of software delivery metrics. These use cases have been observed globally at clients of all sizes and stages of Agile DevOps maturity by the Plandek team.

Figure 22 – most common use case for applying software delivery metrics and analytics

9.1 Track the impact of an Agile/DevOps transformation

Very often it is the CIO who sponsors the roll-out of improved software delivery metrics and analytics. A common reason for this is the need to better track the impact of an ongoing Agile transformation. Such transformations are very time consuming and costly with short 'honeymoon periods'.

As such stakeholders soon start to ask for quantitative evidence that the transformation is delivering improved performance and software delivery outcomes over time.

This requires a meaningful 'balanced scorecard' of delivery metrics for use within the technology delivery capability and for reporting to stakeholders and the C-Suite. Very often these metrics are simply not available with existing reporting plug-ins. As such a specialist BI solution like Plandek is required, which enables the surfacing of a set of delivery, Agile, engineering and DevOps metrics for use internally and externally (see the Metrics section 1).

The impact of tracking and embedding a metrics-led continuous improvement process can be dramatic as teams self-improve over time resulting in material improvements in: velocity, throughput, quality and timing accuracy.

9.2 Deliver high quality software more predictably

This is a very common stimulus for implementing a data-led Agile delivery approach.

As mentioned in the Sprint Metrics section, one of Agile's biggest challenges and criticisms (especially in large organisations) is the inability for Agile teams to predict their output. All organisations will from time-to-time require software to be delivered within a certain timeframe in order to meet the needs of the stakeholder (e.g. due to a seasonal trading period, regulatory requirement or competitive pressure).

In these instances, Agile teams have to provide visibility of timing of increments of output. And in order to be able to provide this visibility, the *dependability* of teams is absolutely vital.

Clients applying metrics in this way may expect to reduce unplanned go-live delays by over 50%.

9.3.Shorten time to market and increase velocity

Pretty well all organisations are constantly trying to deliver valuable software more quickly – indeed that is a core objective of Agile. This can be especially important in companies where speed-to-market is a defining factor of competitive success (or indeed survival).

Applying end-to-end delivery metrics is critical in increasing velocity. Indeed, increasing velocity should not require the hiring of additional resource, but simply by achieved by the removal of bottlenecks from the end-to-end delivery process, in order to reduce Lead Time and increase delivery velocity. Case Study 1 in the Case Studies section shows how a global data and publishing company reduced Cycle Time by 25% across over 2,000 engineers within 6 months by tracking and embedding relevant delivery metrics throughout the delivery organisation.

9.4.Reduce delivery risk and increase visibility in Scaled Agile environments

Whilst Agile software delivery has many benefits it was not originally conceived for application at scale. As such there is a large and growing industry to help organisations apply Agile at scale, with a great variety of Scaled-Agile frameworks and tools.

All of these frameworks try to resolve the paradox at the heart of Agile delivery – that its success is predicated on empowering small teams to be self-determining and in control of their workflow – which is a principle that is very hard to sustain in a large organisation.

Indeed, Agile software delivery is sometimes described as 'controlled anarchy' and large organisations may struggle to ensure that it does not become 'uncontrolled anarchy'! As a result, CIO's often require increased visibility across their Agile delivery teams to better manage delivery risk. These teams may be globally distributed and involve contractors and in-house engineers.

Well implemented delivery metrics not only enable individual teams to adopt relevant metrics and self-improve over time – they also provide the visibility that the technology leadership team needs to better understand where delivery risks may lie at team level across the broader delivery organisation.

9.5.See real difference in performance between teams

Technology leadership are constantly faced with investment and resource allocation decisions. Should I build up my in-house capability? Should I move resource offshore or near-shore? Should I increase my reliance on contractors and SIs?

Very often these decisions are taken on gut feel, politics and anecdotal evidence provided by ley managers.

However, the arrival of robust end-to-end software delivery metrics solutions like Plandek, enable technology leaders for the first time, to take these decisions in a much more objective way – based on balanced scorecard of meaningful delivery metrics tracked over time.

Critics may contend that such comparisons are not possible due to different circumstances between teams (workflows, priorities, engineering challenges). However, metrics solutions are now so complete that these reservations can be conclusively overcome.

9.6 Empower teams to self-improve with analytics over time

A core tenet of Agile software delivery is that teams should respond to customer needs and challenge themselves to constantly improve.

However, very few delivery teams find the time and energy to put in place a robust, metrics-led self-improvement process. This is changing as super-easy metrics solutions like Plandek have become available.

Teams can create customised team-level dashboards with a meaningful set of metrics around which they can self-improve their delivery capability. The metrics soon become embedded in team 'ritual' and targets set and discussed at stand-ups and retros.

Forward-looking technology leadership now recognise the benefits of enabling their teams with such metrics solutions to unleash the power of metrics-led team self-improve over time.

9.7 Remove manual reporting (to C-Suite and within Technology)

Manual reporting, spreadsheets and hand-drafted RAG Reports are a fact of life for many busy delivery teams!

However, the next-generation end-to-end metrics solutions like Plandek, removes this manual reporting effort and places real-time analytics at the fingertips of all who need it within the delivery capability – from the Team Leader, across Delivery, Programme Management, Engineering and Leadership.

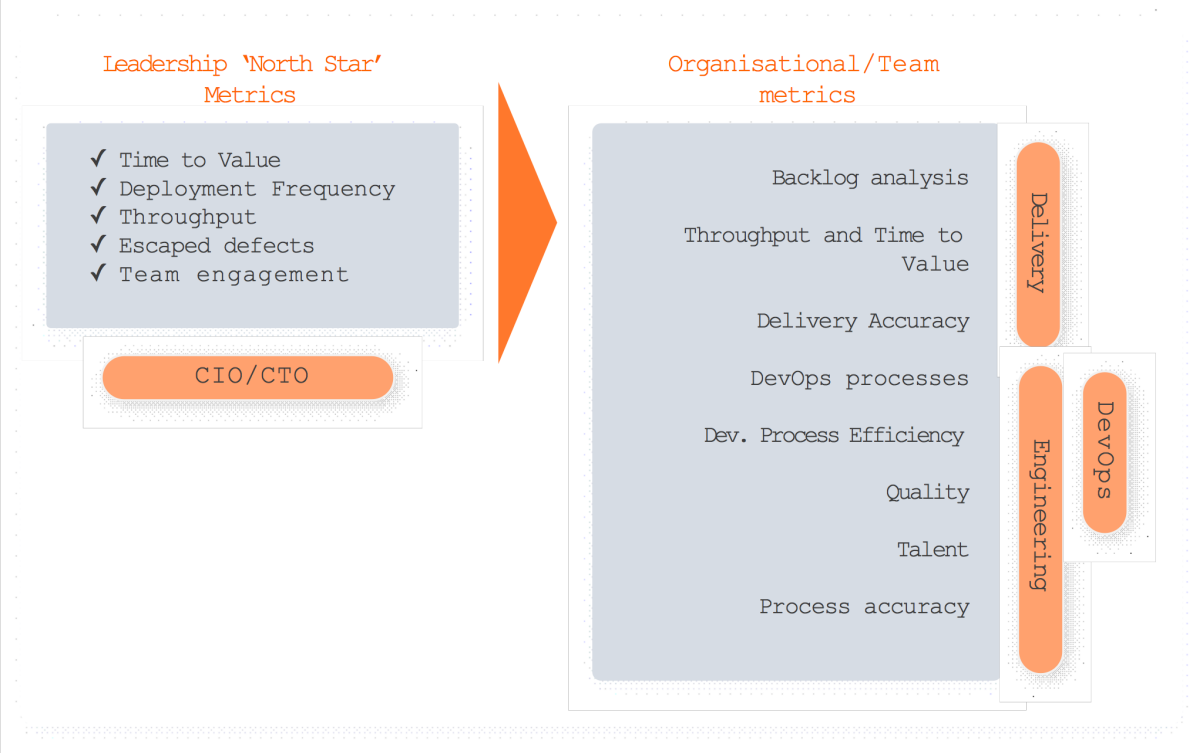
10.The Insight-driven Agile Delivery© (IDAD) framework

1. The principle of Insight-driven Agile Delivery© - creating a hierarchy of metrics that everybody understands

Insight-driven Agile Delivery© (IDAD) is a framework that provides a simple hierarchy of metrics, so that end-to-end delivery metrics are understood and adopted by everyone. This is especially important when team members are working remotely and are under increased pressure to deliver.

As shown conceptually in Figure 23 below, the metrics are self-reinforcing, with team level metrics rolling-up to your overall 'North Star' (delivery health metrics) set by the technology leadership team.

Figure 23 - The Proprietary Insight Driven Agile Delivery© (IDAD) framework



The 'North Star' metrics are carefully selected to reflect the organisation's overall delivery objectives (often related to core Agile objectives) and are the metrics around which technology leadership want the delivery organization to align in order to deliver better over time.

Metrics are then cascade across the delivery organisation, with customised dashboards provided (using Plandek) to surface a balanced scorecard of metrics owned by teams and key managers, covering the end-to-end software delivery process.

The key is that these team metrics should be leading metrics that are deterministic of improving the process (and improving the North Star metrics) — rather than lagging metrics that simply "look in the rear-view mirror".

So, what is a sensible set of metrics to adopt?

In our view, metrics should always reflect the client context and objectives. Hence IDAD is intended as the basis from which clients can build their own bespoke metrics sets that closely mirror their own specific objectives.

Many organisations will take the IDAD metrics listed in this section as a good place to start. And we would agree, as the IDAD metrics ensure that you continue to deliver against the most central Agile principles at a time of rapid change and stress.

However, there are a variety of commonly used metrics setting approaches - such as OKRs (Objectives and Key Results) or GQMs (Goal, Metric, Question)

as popularised by Victor Basili which can also be applied to define a bespoke metric set.

In our view, whichever route you take is very much up to you, but it is the discipline of tracking and managing to metrics (that reflect core Agile principles) that is critical - so that when the chips are down, everyone across your organisation is focused on the things that really matter.

10.2 Suggested 'North Star' metrics

We are not trying to reinvent the wheel with IDAD – it simply offers a set of leading metrics that track your ability to continuously improve against the central Agile principle of software delivery.

These suggested metrics summarise your "Agile health" – your ability to deliver software effectively, despite the constraints. They are meaningful when tracked over time at an aggregate level – and give your whole organisation a simple set of metrics around which to align.

1. *Time to Value* – the core Agile metric tracking how "early" you are delivering value for customers. Measured from the beginning of the development process through to deployment.
2. *Deployment Frequency* – key metric of how "continuously" you are delivering
3. *Throughput* – how much value you are delivering. There are many ways Agile teams will measure this, but Story Points, Value Points or Tickets Completed are a common place to start
4. *Defect Density* – a key measure of your ability to continue to deliver high quality software, commonly measured by the ratio of stories delivered to escaped defects (or production defects)
5. *Team Engagement* – our favourite metric in the 'new normal' world of remote working. Best collected via polling on collaboration hubs like Slack and in our view the key leading indicator of delivery health, as software delivery is absolutely dependent on your team struggling with the pressures of the current environment.

10.3 Cascading IDAD metrics - our top 5 metrics for managers and teams in the new normal world

The power of metrics is realised if metrics are vocally sponsored by leadership and are then cascaded across the organisation to the key functions, programmes and Agile teams (squads) responsible for software delivery.

Our top 5 team metrics to improve your overall delivery capability are shown in the table below.

Deployment Frequency	Critical DevOps metrics such as *Deployment Frequency* which confirm our ability to maintain the continuity of our delivery

Flow Efficiency	Development efficiency metrics such as *Flow Efficiency* (the % of time tickets spend in an active versus inactive status) which are likely to be negatively impacted as teams move to unfamiliar remote working
Cycle Time and Lead Time	*Cycle Time* and *Lead Time* - the critical measures of time to value, which are also likely to suffer in ties of change and stress
Completion Rate	*Completion rate* which measures our ability to deliver our sprint goals over time. This is highly likely to be negatively impacted in the new normal world and will then adversely affect the predictability of delivery timing
Engineer Morale Score	*Engineer morale score* – (measured in Plandek via polling using collaboration tools like Slack). This is a critical measure in the new normal world as people adjust to the often unfamiliar set of circumstances that they find themselves in.

11. Embedding a culture of metrics-led team continuous self-improvement

The IDAD framework described in Section 10 is deigned to help organisations who may be moving to a metrics-led delivery culture for the first time, select and cascade out a meaningful set of end-to-end delivery metrics.

This section considers in a bit more detail who such metrics might be embedded into team culture, in order to drive improvement over time.

This is a classic change management challenge that is the bread-and-butter of SIs like Accenture and Deloitte. However, the Plandek Customer Success Team has unique experience globally in this very particular challenge.

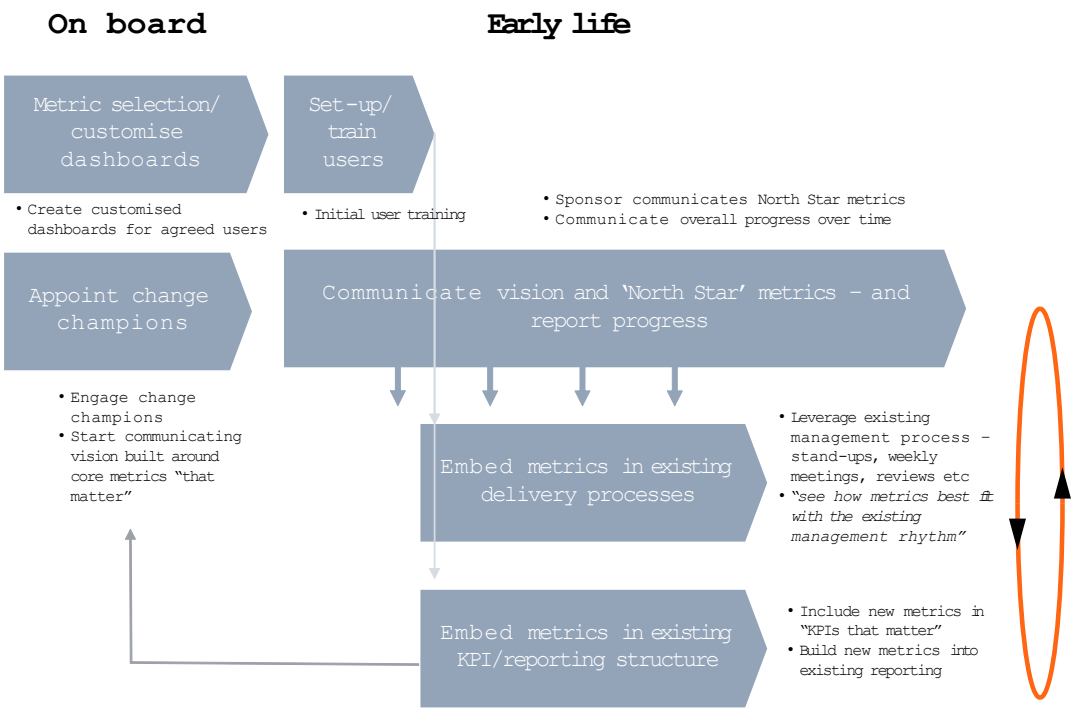
As shown in Figure 25, embedding a continuous improvement programme across a busy delivery organisation is not a trivial task. It requires:

1. Strong leadership sponsorship
2. A well-defined hierarchy of metrics (see IDAD in Section 10)
3. A means of surfacing those metrics in customisable dashboards available at all levels within the delivery organisation; and
4. Strong sponsorship and ownership at Scrum Master and Team level.

As per point (4), the Plandek Customer Success team always works closely at team level to help Team Leads embed metrics in the 'rhythm' of their workflow, so that the metrics that they have selected as appropriate (and in keeping with the overall North Star metrics adopted by technology leadership) become 'second-nature' and are embedded in their daily, weekly and sprint processes. Targets are set and teams embed metric discussion in their stand-ups, sprint retros and other key communication points.

When this happens, it unleashes a powerful continuous-improvement effect that quickly translates into significant and lasting performance improvements at the organisational level. (See Case Studies).

Figure 25 – typical methodology to embed a metrics-led continuous improvement culture



CASE STUDIES

12. Case study 1 – Reducing Cycle Time by 25% at a global data business

The client context: This high profile, multi-national data and publishing business use Plandek as a key element of their Value Stream Management across their global software delivery teams with over 2,000 engineers in multiple locations. Plandek provides the metrics and reporting to underpin their OKR (Objectives and Key Results) process within the global delivery organisation. As an example, Cycle Time was identified using Plandek as a key opportunity area for improvement and a **specific OKR was created to reduce Cycle Time by 25% in 6 months** during 2020.

Cycle Time was deemed critical due to the increasing pressure to deliver new features rapidly in a sector where speed-to-market is a critical differentiator.

Using a variety of delivery and engineering metrics available within the Plandek platform, including 'Mean Time to resolve Pull Requests' and 'Flow Efficiency', the teams **drove a number of process improvement initiatives** and saw month on month reductions in Cycle Time resulting in an average 25% reduction in Cycle Time between January and June 2020.

5 Key Takeaways

1. A **continuous improvement initiative** underpinned by a **simple set of 'North Star' metrics that teams understand and trust can deliver rapid, sustainable and significant improvement** in software delivery outcomes at scale

2. **In large scale delivery environments, OKR provides an effective framework** to prioritise a set of **simple targets for improvement** (such as a 25% improvement in Cycle Time). And **Plandek is an ideal BI tool to provide the necessary end-to-end software delivery metrics** to underpin the collective effort to deliver the OKR targets set

3. Using Plandek, **four metrics were found to directly impact Cycle Time across multiple teams: Flow Efficiency** (which looks at the proportion of time tickets spend in an 'active' versus 'inactive' status); **Mean Time to Resolve Pull Requests** (hrs); **First Time Pass Rate** (%); and **Story Points Ready for Development**

4. **Plandek was embedded in teams' management processes** (e.g. standups, sprint retros) to track and manage to these four determinant metrics, **with the result that average Cycle Time across workstreams was reduced by over 25% over a 6 month period.**

5. **This was only possible as teams trust the quality of the metrics/ analytics** as Plandek enables them to see the 'provenance' of the metric (how it is calculated) and to configure metrics to match their precise team circumstances (via Filtering functionality)

Creating a hierarchy of simple metrics that everyone understands

Plandek can surface a myriad of metrics. The Plandek Customer Success team worked closely with the client to identify a simple set of 'North Star' metrics, (selected from this broader potential metrics set), around which to set their delivery goals.

The 'North Star' metrics were carefully selected to be meaningful when aggregated and illustrative of effective Agile software delivery:

'North Star' Metric	Agile software delivery objective
Cycle Time	Early and continuous delivery
Deployment Frequency	
Throughput (Delivered Story Points & Value Points)	Delivery of value
Sprint Target Completion	Dependability of value delivery

These North Star metrics were adopted by the technology leadership team as key priorities within an OKR (Objectives and Key Results) framework.

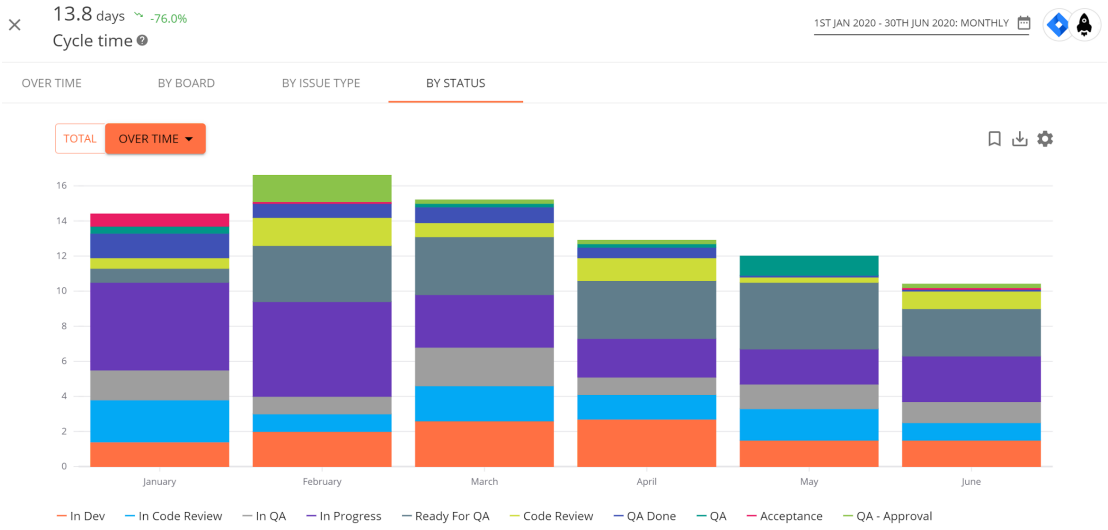
Setting Cycle Time as an OKR target

As Time to Value was identified as a key priority (and opportunity for improvement), **an OKR target was agreed to reduce Cycle Time by 25% over 6 months in H1 2020.**

The Plandek network of dashboards allowed each team to closely analyse their own Cycle Time and understand where in the Cycle there was an opportunity to drive down time to value.

As per figure 26 below, the Plandek Cycle Time metric view allowed teams to understand time spent in each ticket status within the development cycle. The flexible analytics capability and powerful filtering allows analysis by Status, Issue Type, Epic (and any other standard or custom ticket field) all plotted over any time range required.

Figure 26 – Example Plandek Cycle Time metric view



Tracking and improving key metrics that drive Cycle Time, to deliver the OKR

Reducing Cycle Time by 25% is an aggressive target, which if delivered effectively, drives very significant business benefit as software is delivered more rapidly without additional delivery resource allocation (or impact on quality).

Working with the Plandek Customer Success team, Plandek was used by scrum teams to identify key determinant metrics that would have the biggest impact on reducing Cycle Time without impacting quality or requiring additional resource allocation.

Analysis showed four metrics which could unlock significant shortening of Cycle Times across almost all scrum teams. These were:

- **Flow Efficiency** (which looks at the proportion of time tickets spend in an 'active' versus 'inactive' status)
- **Mean Time to Resolve Pull Requests (hrs)**
- **First Time Pass Rate (%).**
- **Story Points Ready for Development.**

Each scrum team and related Scrum Masters and Delivery Managers updated their Plandek dashboards to surface these critical metrics, so that they could be tracked and analysed in daily stand-ups, sprint retrospectives and management review meetings.

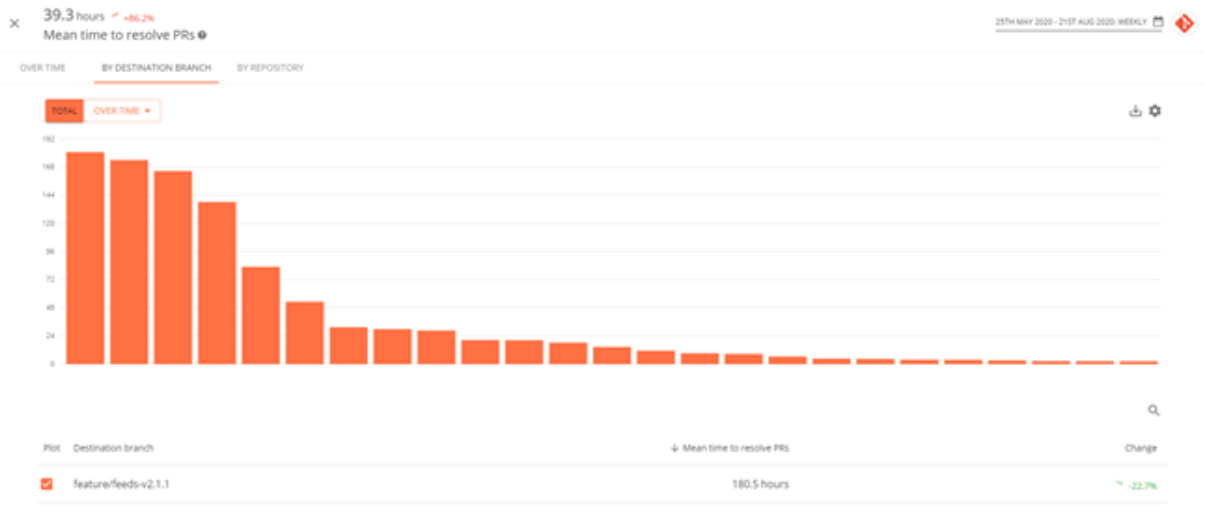
The Flow Efficiency analysis enables Team Leads to isolate and analyse each 'inactive' status in the workflow and consider if there is scope to reduce or eliminate it. The analysis shows the relative size of each 'inactive' status opportunity in terms of time spent in the inactive state and volume of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

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 100 hours not uncommon.

Plandek enables drill-down to understand variances by code repository and destination branch (see Figure 27 below). This enabled quick identification of the biggest bottlenecks and targeted intervention, with the result that MTRPR was reduced dramatically (by <80% in some squads) and by an average of c50%. This has a very significant impact on overall Cycle Time.

Figure 27 – Example Mean Time to Resolve Pull Request metric within Plandek dashboard



First Time Pass Rate (FTPR) was another key metric in driving the 25% Cycle Time improvement achieved over the six month period. It proved to be a

popular metric at team level as high FTPR not only increases velocity (and reduces QA burden), but is symptomatic of a productive relationship between BAs, engineers and QA - with well managed backlogs, well defined tickets/ requirements and hence a smoother flow of tickets through the development process.

Drill-down within the Plandek "Explore" functionality shows variations in FTPR by Board, ticket and individual within the team.

Figure 28 – First Time Pass Rate example metric within the Plandek dashboard

Effective analysis of teams' backlog proved to be a fertile area for identifying bottlenecks that reduced velocity and adversely affected Cycle Time.

Teams with well managed backlogs (i.e. with at least 2 sprints worth of tickets prepared and ready to progress), significantly reduced their Cycle Times. As such, the simple metric of **Story Points Ready for Dev** was a key metric in increasing velocity across the majority of teams. The powerful Filter functionality within Plandek enables teams to identify and track relevant ticket types to ensure accurate analysis.

Metrics led Continuous Improvement in software delivery – buy-in and trust

The experience at the client showed the power of applying a metrics-led philosophy across a scaled Agile software delivery capability. Cycle Time was reduced by just over 25% over a 6 month period in H1 2020, thereby meeting the OKR set by the technology leadership team.

Key factors in the success of the approach included:

1. The identification and communication of a simple delivery goal in keeping with the underlying Agile delivery approach (a reduction in Cycle Time)
2. The use of Plandek to surface that metric in real time at all levels within the delivery hierarchy (across Board, team, workstream, PI, tribe etc)

3. Collective buy-in and trust in the metrics from Team Leads upwards. This was critical and was made possible as a result of the total transparency of Plandek metric presentation.

Experience shows that if Team Leads cannot see exactly how metrics are calculated and that they reflect their team's context – they will question and ultimately reject the metrics – especially if the metric appears erratic or heavily negative.

Plandek is unique in its ability to show the 'provenance' of each metric and to allow individual teams to configure each metric in the way that reflects their particular circumstances, using the powerful Filter functionality. This is ultimately critical in the overall success of the initiative.

13. Case Study 2 - Building an insight driven delivery organisation with Plandek: using Plandek to reduce Cycle Time by 75% and increase deployment frequency by 15%

The client

The client is one of Europe's technology-led travel success stories operating in twelve countries across Europe and in the US.

The client values an insight-led approach to software delivery and uses Plandek as a key element of its DevOps Value Stream Management across all its software delivery teams. Plandek's customised dashboards are used across the delivery organisation to provide metrics, analytics and reporting, to underpin a robust Continuous Improvement process. The process is led by individual Team Leads and managed and sponsored by technology leadership.

This metrics-led approach to continuously improving the software delivery process has been highly successful, with major improvements seen in key metrics over the last 24 months.

4 Key Takeaways

- A continuous improvement initiative underpinned by a simple set of 'North Star' metrics that teams understand and trust can deliver rapid, sustainable and significant improvement in software delivery outcomes at scale
- Plandek is an ideal BI tool to provide the necessary end-to-end software delivery metrics to underpin the collective effort to deliver rapid improvement in delivery outcomes
- Over the past 24 months, using Plandek to underpin a robust continuous improvement process, the client has:
    - Reduced Cycle Time by 75%
    - Reduced hot-fixes in Prod by 54%
    - Doubled commit frequency by Engineers
    - 15% increase in deployments per day (per pipeline)

- The improvements seen were only possible as teams trust the quality of the metrics/analytics as Plandek enables them to see the 'provenance' of the metric (how it is calculated) and to configure metrics to match their precise team circumstances (via Filtering functionality)

Creating a hierarchy of simple metrics that everyone understands

Plandek can surface a myriad of metrics. The Plandek Customer Success team worked closely with the client to identify a simple set of 'North Star' metrics, (selected from this broader potential metrics set), around which to set their delivery goals.

At the client, the 'North Star' metrics were carefully selected to be meaningful when aggregated and illustrative of effective Agile software delivery:

'North Star' Metric	Agile software delivery objective
Cycle Time	Early and continuous delivery
Deployment Frequency	
Throughput (Delivered Story Points & Value Points)	Delivery of value
Sprint Target Completion	Dependability of value delivery

These North Star metrics were adopted by the technology leadership team as key priorities and cascaded across the delivery organisation - along with a set of determinant metrics that drive improvement in these critical North Star metrics (KPIs).

As such, key players across the delivery organisation have their own Plandek customised dashboards with the determinant metrics relevant to their area. These key players include Team Leads, Delivery Managers, Scrum Masters and DevOps/Engineering Managers.

The sections below consider the most popular determinant metrics used by the client to drive continuous improvement in the North Star KPIs.

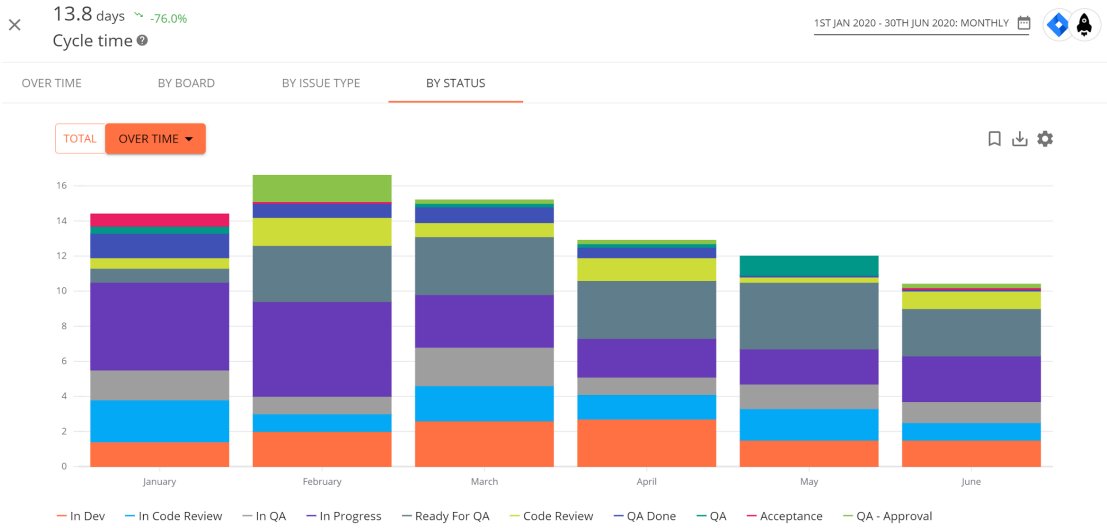
Driving continuous improvement in Cycle Time

As a 'North Star' metric, *Cycle Time* was quickly adopted as a key focus for delivery teams. The Plandek network of dashboards allowed each team to closely analyse their own Cycle Time and understand where in the Cycle there was an opportunity to drive down time to value.

As per figure 29 below, the Plandek Cycle Time metric view allows teams to understand time spent in each ticket status within the development cycle.

The flexible analytics capability and powerful filtering allows analysis by Status, Issue Type, Epic (and any other standard or custom ticket field) all plotted over any time range required.

Figure 29 – Example Plandek Cycle Time metric view



Working with the Plandek Customer Success team, Plandek was used by scrum teams to identify key determinant metrics that would have the biggest impact on reducing Cycle Time without impacting quality or requiring additional resource allocation.

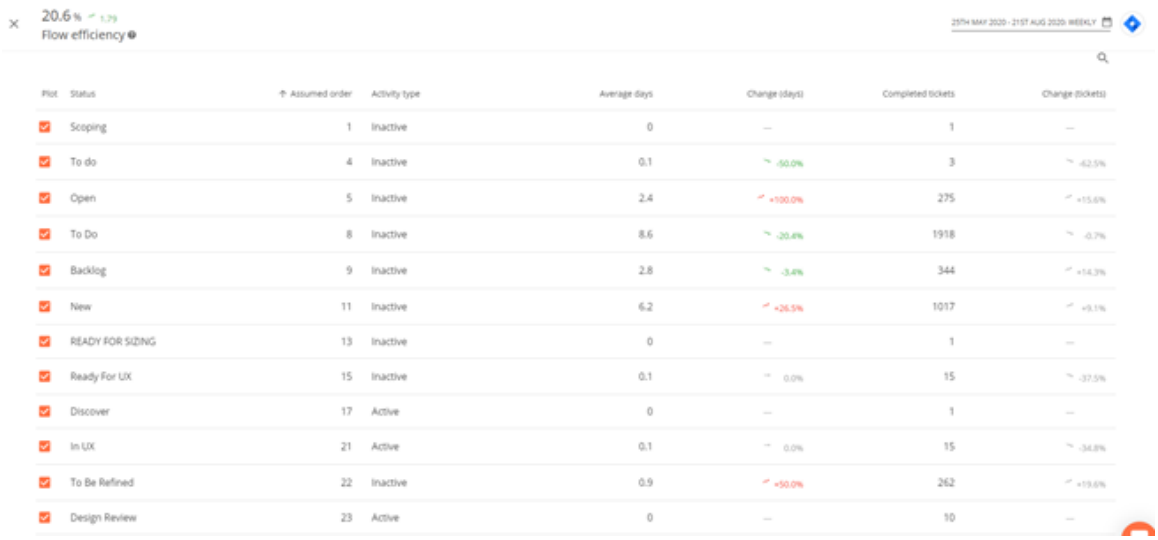
Analysis showed three metrics which could unlock significant shortening of Cycle Times across almost all scrum teams. These were:

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

Each scrum team and related Scrum Masters and Delivery Managers updated their Plandek dashboards to surface these critical metrics, so that they could be tracked and analysed in daily stand-ups, sprint retrospectives and management review meetings.

The Flow Efficiency analysis (see Figure 30 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 30 - Example Flow Efficiency metric within Plandek dashboard

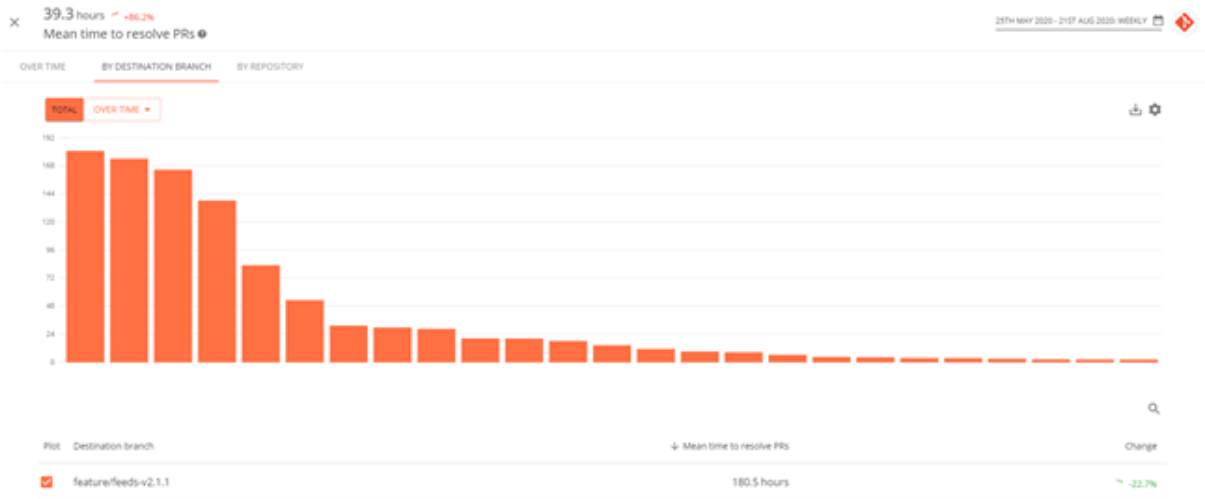


Typical opportunities to remove inactive bottlenecks included time spent with tickets awaiting definition (e.g. Sizing) and tickets awaiting QA. Where waits for QA were considered excessive, Delivery Managers reconsidered QA resource allocation by team.

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 31 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 31 – Example Mean Time to Resolve Pull Request metric within Plandek dashboard



Increasing deployment frequency and reducing failed builds

In keeping with Agile principles, *Deployment Frequency* is a 'North Star' metric and hence a consistent focus for the delivery organisation at the client. Plandek's end-to-end view of the delivery process enables delivery teams to closely track deployment frequency and track and manage the bottlenecks that may be slowing frequency of deployments.

DevOps practitioners can track a range of metrics including: *Number of Builds, Build Failure Rate, Deployment Cycle Time* and *Flakiest Files* (which identifies fragile source code files in your codebase which can be targeted for refactoring.)

The client i**ncreased deployments per day** (per pipeline) by 15% through a better understanding of the root-cause of Build Failures and Deployment Cycle Time using Plandek.

Increasing throughput and value delivered

Delivery Team Leads and Managers adopted a range of determinant metrics that help track and drive the delivery of value (see Figure 5).

These included *Stories Delivered by Epic, Lead Time for Stories and Epic, and Delivered Value Points.* And *Mean Build Time and Deployments by Pipeline* were also used to track and improve the rate of deployment of value.

Teams use the Plandek drill-down functionality (and ability to review individual tickets within Jira) to continually review progress and unlock bottlenecks.

Improving quality of delivery: reducing hot-fixes in production

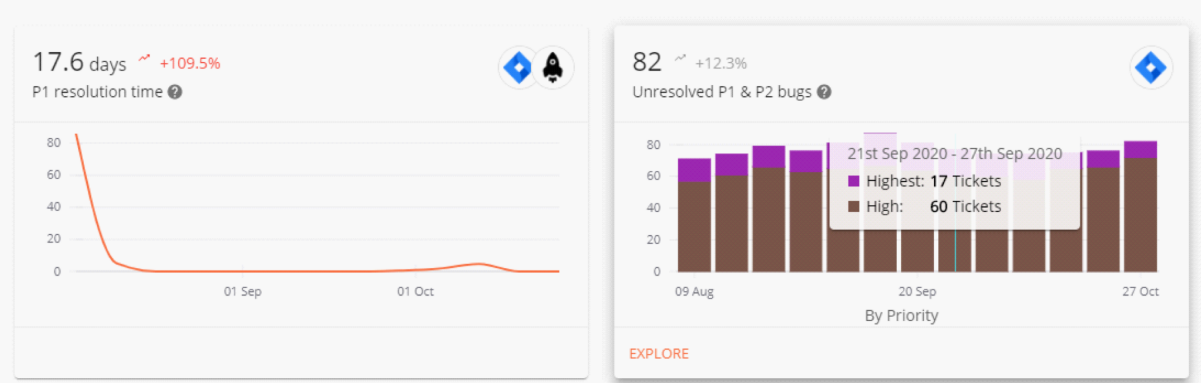
A key thrust of the client's insight-led approach, is to use trend data to quickly identify where improvements can be made. Quality is a consistent focus - both the security and quality of the delivery process itself and the quality of the software delivered.

Historical data in Plandek revealed trends in the overall *Hot Fix Rate* (sample data illustrated below as *Escaped Defect Rate)* and the opportunity 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.

The net result was a more disciplined approach to bug resolution across teams with the result that hot fixes in production were reduced by 54% over a 12 month period.

Figure 32- Example quality metrics: P1 Resolution Time and Unresolved Bugs



The client adopts a broad view of 'quality' to include both the software output and the quality and security of the delivery process itself. As such teams also track and manage *Commits without a Pull Request* and *Commits without a Ticket Reference*. See Figure 33.

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 33 - Example delivery process quality metrics

Metrics led Continuous Improvement in software delivery – buy-in and trust

The experience at the client shows the power of applying a metrics-led philosophy across an Agile software delivery capability. As described, Team Leads led significant improvement across a range of critical Agile metrics including: *Cycle Time; Escaped Defects, Deployment Frequency* and *Commit Frequency* by engineers.

Key factors in the success of the approach included:

- The identification and communication of a simple set of 'North Star' metrics around which the delivery organisation aligns
- The use of Plandek to surface determinant metrics in real time at all levels within the delivery hierarchy (across Board, team, workstream, PI, tribe etc)
- Collective buy-in and trust in the metrics from Team Leads upwards. This was critical and was made possible as a result of the total transparency of Plandek metric presentation.

Experience shows that if Team Leads cannot see exactly how metrics are calculated and that they reflect their team's context – they will question and ultimately reject the metrics – especially if the metric appears erratic or heavily negative.

Plandek is unique in its ability to show the 'provenance' of each metric and to allow individual teams to configure each metric in the way that reflects their particular circumstances, using the powerful Filter functionality. This is ultimately critical in the overall success of the initiative.

14. Case Study 3 - Using Plandek to improve the dependability of teams to deliver software more predictably: Increasing sprint accuracy by 15% at a European fintech business

The client context: This highly successful provider of SaaS accounting software use Plandek as a key element of their Value Stream Management across their distributed software delivery teams in multiple locations. Technology leadership and the delivery teams themselves use Plandek's customisable dashboards to track and continuously improve end-to-end delivery metrics.

A key focus for the client is the dependability of the scrum teams. Enabling teams to accurately meet sprint goals (over a two week period) is seen as a key building block to dependable software delivery. With multiple teams working on complex product workstreams over many weeks – individual team's failure to regularly meet sprint delivery goals would quickly result in highly unpredictable programme increment outputs.

Using a variety of delivery and engineering metrics available within the Plandek platform, the teams **drove a number of process improvement initiatives** and improved sprint delivery accuracy from c70% to >80% over a six month period (an increase of c15%).

Introduction – the client context

The client operates a scrum Agile software delivery capability onshore and offshore across multiple workstreams.

The company is metrics-led and adopted Plandek as the group-wide solution to surface end-to-end software delivery metrics across all teams, in order to greatly improve visibility across teams and thereby:

- reduce software delivery risk (improve delivery dependability)
- improve software delivery productivity and quality
- demonstrate the success of their Agile transformation with a balanced scorecard of improving metrics over time.

Plandek was first adopted in early 2020 and the Plandek Customer Success team have since worked closely with the client to help them create and

embed customised dashboards for all teams (squads), Delivery Managers and technology leadership.

Scrum team dependability, reflected in their ability to accurately deliver sprint goals over a two week sprint period, was quickly identified as a high priority as some critical delivery milestones (linked to key commercial business objectives) were upcoming.

The question was therefore raised: "Which metrics should we look at during our sprint retrospectives and daily stand-ups to help ensure that we meet our sprint goals?"

For Scrum teams, there are a few key areas that determine both short and long term success. The client's aim was not only to meet their current sprint goals but also to build and maintain healthy patterns of work and collaboration that will lead to future success.

Working alongside the Plandek Customer Success team, the key questions asked about sprint performance were:

- Are we able to meet our commitments/goals reliably?
- Is our work flowing smoothly throughout the sprint?
- Are there any risks emerging that may impact our ability to meet our sprint goals?
- Has this sprint improved our overall delivery performance?

Below we explore each of these questions further and outline how the client used Plandek to track some simple sprint metrics to help answer the questions and improve the dependability of all their scrum teams.

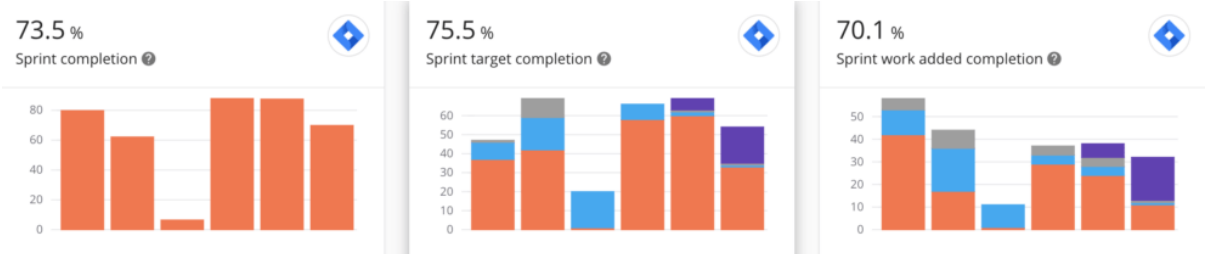
Meeting Sprint Commitments

The client selected three key metrics from the Plandek metric library to track teams' sprint overall accuracy: *Sprint Completion*, *Sprint Target Completion*, and *Sprint Work Added Completion*.

Perhaps the most important of the three, *Sprint Target Completion* looks at the scope you agreed to during sprint planning and tracks how much was completed, showing you how effective the team is at establishing the right priorities and delivering them.

Sprint Work Added Completion focuses only on work that was added to a sprint after it started (which is a very common problem for scrum teams), whilst *Sprint Completion* looks at the whole picture, regardless of whether work was planned for the sprint or added afterwards.

Figure 34: Three key sprint completion metrics



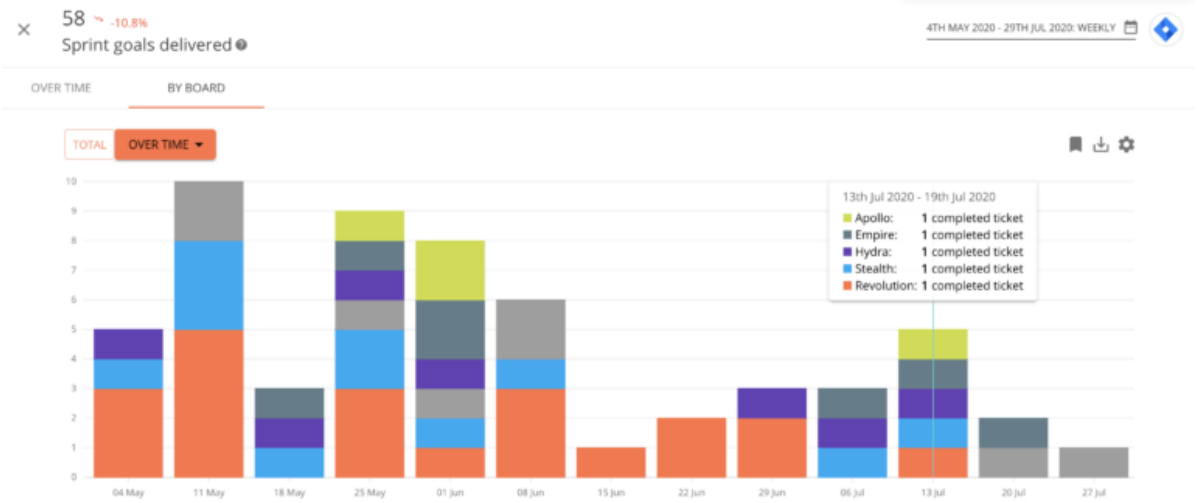
In our experience across multiple clients, **Sprint Target Completion rates lower than 80% can start to cause serious dependability problems, especially in Scaled Agile environments with many teams and Programme Increments to navigate.** Indeed, predicting the delivery status at the end

of a single PI becomes extremely difficult if multiple teams are involved and many are consistently not meeting their Sprint Target Completion.

This can be for many different reasons, but very often is a result of consistent problems with ticket sizing, which can be worked on with objective review in sprint retros.

Alternatively, if you use Sprint Goals to define specific objectives above and beyond the work planned in the Sprint, you may also find this to be a useful metric for your retrospectives.

Figure 35: Example Sprint Goals Delivered Metric



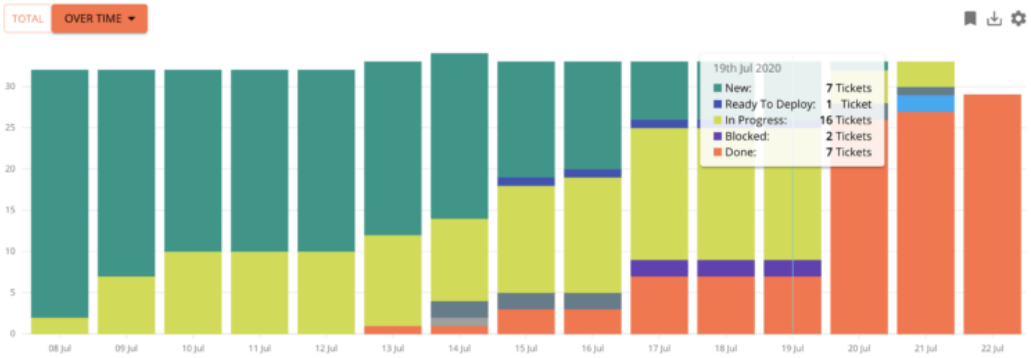
Delivering efficiently within a sprint

Client teams also adopted a number of 'determinant' sprint metrics that together can significantly improve overall sprint accuracy (as tracked by Sprint Target Completion). Clearly in sprints, you only have a couple of weeks to deliver a specific scope of work, so it's critical that:

1. workflows smoothly throughout the sprint,
2. bottlenecks/delays are spotted and addressed immediately, and
3. user/PO feedback is provided to the team as quickly as possible so any issues can be resolved within the sprint.

The client selected *Ticket Timeline* (see below) from the Plandek metric library, to track how their work was flowing throughout the Sprint, and easily spot any delays or bottlenecks emerging over the two week period, that may have potentially put their commitments at risk. As such, it proved an incredibly useful metric in sprint retros and daily stand-ups.

Figure 36: Example Ticket Timeline analysis



Identifying risk and mitigating it

Whilst Ticket Timeline above proved a powerful way of surfacing the impact of risks on delivery, there are a number of other metrics that we recommend that combat some common challenges that teams face. The client Team Leads adopted a range of these metrics (based on their own preferences) to closely track performance over time.

Moving goalposts

Whilst the client embraced changing priorities, too much change within an active sprint will compromise a team's ability to deliver effectively (and should raise questions on the planning process). With *Ticket Scope*, client Team Leads tracked any key tickets being added or removed from a sprint, which was great for retros and in stand-ups.

Figure 37: Example Sprint Scope graphic

Figure 37 shows a typical sprint with a manageable number of tickets being added and removed throughout the duration of the sprint, reflecting the agility of a mature scrum team. However, it was common for multiple tickets to be added later in the sprint with the result that 'agility' becomes 'conflicting priorities and potential inefficiency'.

Unplanned bugs

New bugs/defects, particularly those from Production, can derail teams very quickly. It's important to track the arrival of new bugs that can side-track teams. Even if bugs are not immediately resolved, the triage process

may (and often does) distract teams from their core focus on delivering sprint work.

As a result, client teams filtered by critical bugs (e.g. P1 and P2), as well as distinguishing between bugs originating from production vs your "QA/UAT" process.

Figure 38: Example timeline of unplanned bugs

Keeping the 'big picture' in mind

We recommend that every organisation has a set of 'North Star' metrics that they use to measure their overall delivery effectiveness and agility. The client adopted this approach with a simple set of four agile metrics championed by technology leadership to give the entire delivery organisation a set of key metrics around which to align.

The client found that sprint retrospectives provided a great opportunity to reflect on how the work delivered in that sprint has contributed to the overall progress against these 'North Star' metrics.

Two of these 'North Star' metrics were *Lead Time* and *Cycle Time*, chosen by technology leadership as they reflect one of Agile's core values: the "*early and continuous* delivery of valuable software".

During each team's retrospective, they reflected on how the sprint's deliverables had impacted the trend over time and examine where there are opportunities to improve in future sprints.

Figure 39: Example graphic showing Cycle Time variance over time

The Cycle Time metric in Figure 39 refers only to the development cycle time and therefore excludes additional time taken to integrate, test and deploy to live. This more complete view of the end-to-end delivery process is reflected in the Lead Time metric which is ultimately the more representative measure of true agility, though it is not so suited for a scrum team as it takes into account delivery stages beyond the scrum team's control.

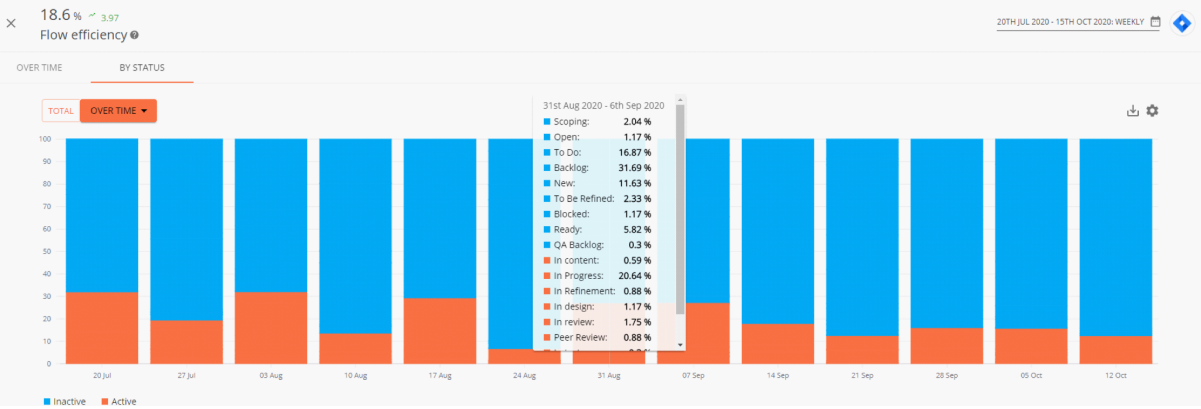
Where can we improve?

The client also adopted Flow efficiency as a 'North Star' metric as it requires teams to focus on areas where process inefficiencies may lie that adversely affect Lead and Cycle Times (and hence velocity).

Teams could see precisely where they were spending the most inactive time, e.g. 'Awaiting QA', 'To Do', 'Awaiting sign-off', and then agree on some focused actions to reduce this *waste* in future sprints.

It is not uncommon for teams to have a Flow Efficiency of less than 20%, meaning that over 80% of the team's Cycle Time is taken up with tickets in potentially avoidable 'inactive' statuses.

Figure 40: Example Flow Efficiency graphic



Bringing it all together

We believe that the metrics above should form the backbone of any team's sprint retrospectives and indeed they were very effective in increasing the client's overall sprint accuracy measured by their Sprint Target Completion - increasing Sprint Target Completion from c70% to >80% over the six month period of continuous improvement.

However, they are not the only metrics you may want to consider. Teams will face different challenges over time and may have different self-improvement initiatives in flight during their sprints, so any metrics you are using to track these should also be included.

In terms of what you might have in your retrospective versus stand-ups, we believe the answer is pretty simple: the same! If the metrics you chose for a retrospective reflect success for your team, then the stand-up is merely a good opportunity to check your progress against your targets so that you can ensure success, intervening if and where possible.

About the author - Charlie Ponsonby

Charlie started his career as an economist in the developing world, before moving to Andersen Consulting. He was Marketing Director at Sky until 2007, before leaving to found and run Simplifydigital in 2007. Simplifydigital was three times in the UK Sunday Times Tech Track 100 and grew to become the UK's largest broadband comparison service. It was acquired by Dixons Carphone plc in April 2016.

Charlie co-founded Plandek with Dan Lee in October 2017. Plandek as an end-to-end delivery metrics analytics and BI platform, to help technology teams better manage delivery risk and improve their delivery effectiveness.

It mines data from toolsets used by delivery teams (such as Jira, Git and CI/CD tools), to provide end-to-end delivery metrics to optimise software delivery forecasting, risk management and delivery effectiveness. Plandek is used by clients globally including Reed Elsevier, Jato Dynamics, Autotrader.ca and Secret Escapes.

About the author - Will Lytle

Will is the Director of Customer Success at Plandek and is passionate about helping his clients build high performing, motivated delivery teams. He works closely with them to identify their biggest delivery challenges, form meaningful objectives, and find the right metrics to drive success.

Will joined Plandek in 2019 from Deloitte, where he specialised in digital transformation and leading cross-functional delivery teams. He has over 15 years of global experience built on digital delivery expertise, operating models, developing talent, and helping businesses shape and deliver technology-enabled business transformations.