KUGLER MAAG CIE

ETAS

DRIVING EMBEDDED EXCELLENCE

# Software Delivery Performance KPIs in Automotive

by
Rainer Dammers, ETAS
Sergej Weber, Kugler Maag Cie

## Introduction

Measuring performance in software delivery is difficult. This is in parts because, unlike manufacturing, inventory is invisible and software development produces unique artifacts [1]. Also, in software engineering, the sequence of work does not follow the same linearity as that of a production line: design and delivery activities – especially in Agile software development – happen simultaneously.

To steer the delivery process in the right direction, businesses selling software products collect a growing number of metrics – the more the merrier. Metrics are so called lagging indicators which measure outcomes and results [2]. While metrics appear informative, they may not matter intrinsically. Their purpose is to support Key Performance Indicators (KPIs). KPIs in turn support the overall business strategic goals and objectives [3]. KPIs are useful only if they are designed to answer business questions, such as »How efficient does my organization operate?«, »Will we deliver to our commitment?« or »How quickly/flexible can my organization adapt to changing market demand and/or satisfy customer needs?« After all, Peter Drucker taught us that »The most serious mistakes are not being made as a result of wrong answers. The truly dangerous thing is asking the wrong ques-tions.«

The problem is, organizations can measure almost anything, but they cannot pay attention to everything. Thus, modern software organizations will focus on KPIs that support their specific business goals the best, allowing them to use those performance indicators as an effective tool for continuous learning and improvement. The goal should be to gather feedback continuously to adapt to an everchanging market as quickly as possible. For this, an efficient delivery pipeline is a prerequisite [4].

As soon as organizations have defined their software delivery performance in a measurable way, they can make evidence–based decisions about how to improve the performance of the delivery process of their software-based products and services. Furthermore, they can compare and benchmark teams against the larger organizations and against the industry as a whole.

So, the big question: Which KPIs are most suitable for measuring software delivery performance?

While a review of the literature of the recent years reveals a plethora of different KPIs quantify and qualify software delivery performance, in this whitepaper we focus on the KPIs proven effective in a multi-year research published in the seminal book »Accelerate« by Forsgren et al. and apply them to embedded development.

Essentially, these four adapted KPIs [5] suggest a link between organizational performance and software delivery performance in the embedded industry:

- Overall Lead Time
- Deployment Frequency / Delivery Frequency
- Mean Time To Recover / Mean Time To Repair (MTTR)
- Change Fail Rate (CFR)

After a quick note about KPIs, we will discuss the three highlevel types of product classes and the four adapted Software Delivery KPIs, provide guidance on how to work with these KPIs and end this whitepaper reviewing a use case from ETAS, a German company and subsidiary of the Bosch Group which designs solutions and tools for the development of embedded systems for automotive and other sectors of the embedded industry.

## A Quick Note About KPIs

Software development efficiency or productivity can be defined as the ratio between the value of the software produced and the effort required to develop it. The measurement of software development productivity or efficiency to date is a story of failure and unintended consequences. While a plethora of technical metrics are collected and presented on a regular basis, very often they measure output rather than outcome [1]:

- Counting lines of code incentives bloat in the software delivered instead of measuring actual productivity or efficiency; this metric is largely discarded as it punishes simplification, refactoring, and reduction of technical debt.
- Capturing developer utilization incentivizes looking busy over maximizing value, actively damages innovation potential and prevents Flow as it causes bottlenecks in the overall value chain.
- Measuring and comparing Story Points misused a powerful planning tool for a metric which then incentivizes inflation of estimates, dilutes planning, and eliminates the value of its intention.

A KPI is a metric, but not every metric is automatically a KPI. While metrics provide raw information about outcomes and results without any context, KPIs are selected to indicate a trend related to a particular concern and can be considered as leading indicators, which originated in economics, where they are defined as a measurable factor that changes before the economy follows a trend. While leading indicators suggest that conditions are favorable for a particular outcome, there is no guarantee [2].

## Product Classes and Software Delivery KPIs

Software can be delivered in different ways depending on its business model and the way the software solution is consumed by users. Here, we differentiate between three high-level types of product classes:

1. Off-the-shelf (OTS) products are brought to customers prepacked and operated by themselves. A common example of an OTS product would be Microsoft Office, where the end user receives the software preinstalled on a new Personal Computer (PC) and/or installs software versions from a data carrier or download image and is solely responsible for configuration and operation of the product. Since almost all devices are connected to the internet nowadays, this product class includes also embedded software which initially is delivered as one package with any device or Electronic Control Unit (ECU) but can now be updated over-the-air (OTA) or by local service as required with new and enhanced version(s) of the software.
2. Off-the-shelf with customization (OTS + custom software) products are products which require customer-specific additions or modifications to provide any value to a customer. A typical example for this kind or product would be SAP which is highly customized in every installation.
3. Software as a Service (SaaS) is a delivery model of IT utility where customers obtain the subscription for a software service without any form of physical delivery. In this case, the full control and responsibility of the operations stay with the SaaS provider. Typical examples of this model are Salesforce, Microsoft Azure, and Amazon Web Services (AWS).

The following KPIs are adapted to the embedded industry to reflect the efficiency of the entire end-to-end value chain of a product, solution or other artifact that generates revenue.

### Overall Lead Time

In general terms, lead time is defined as the time it takes from the end of a previous step of any process until the current process step is completed: Lead Time = Delay Time from Last Step + Process/Cycle Time of the current step. Therefore, Overall Lead Time can be calculated by adding up the lead times for the process steps of which it is comprised.

Overall Lead Time is defined by the time it takes for an element that provides business value to a customer, e.g., a feature to move through the value chain – from the earliest point of capturing the business hypothesis or market requirement to the point of being:

- delivered to a repository or download server from which it can be released to a customer in the case of an OTS software product,
- deployed directly to production environment in case of a SaaS solution.

As depicted in Fig. 1, Overall Lead Time is the sum of the lead times of the exploration phase, the execution/implementation phase, and the delivery/deployment phase. Those phases again are comprised of multiple process steps with their own lead times. For instance, the implementation phase includes feature development and feature completion.

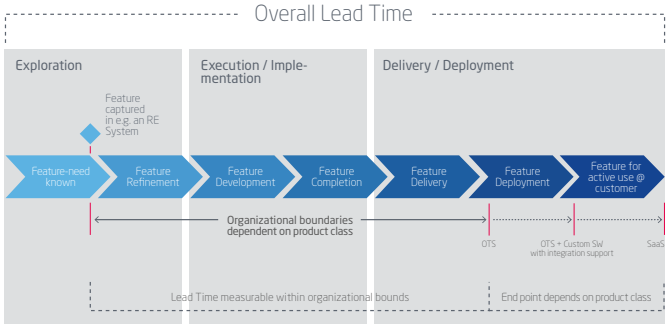


Fig. 1: Overall Lead Time variations

As software development produces unique artifacts, the Overall Lead Time will always vary. In fact, a lead time is not a discrete number, it is a probability distribution. Providing a median regarding delivery performance is not enough. At least two values are important to communicate: the median lead time and a high percentile value. As illustrated in the fictional example in Fig. 2, it takes 20 days for a work item to be delivered on average. Here, the probability to deliver within 18 days is 85%, the probability to deliver within 27 days is 98%. The median value is a delivery of 12 days.

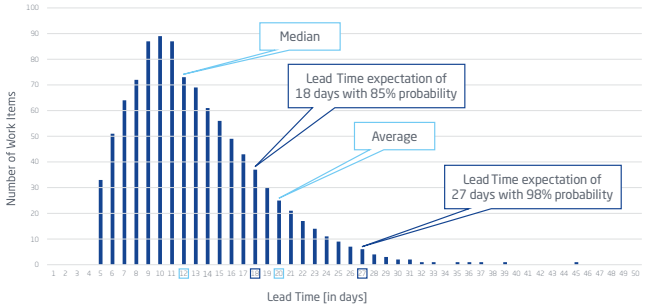


Fig. 2: Lead Time as probability distribution (fictional example)

### Deployment Frequency / Delivery Frequency

Since value is only tangible for the end users when they get to use the item of value, releasing that value at the right point in time is business critical. Deciding when and what to release is a critical economic factor that must be carefully considered. More typically for SaaS solutions, new functionality can be released as soon as it reaches a certain product maturity. More often, releasing is a decoupled, on-demand activity that occurs for specific users at the time they need the functionality or when it makes the most economic sense for the organization.

Deployment Frequency and Delivery Frequency are very related concepts and generally describe the frequency of how often an item provides business value to the customer. For instance, a feature completes the full value chain (as shown in Fig. 1) when:

- Delivery Frequency describes how often an item of value is made available on a delivery system, such as a repository or download server, from which it can be released anytime based on a business decision.
- Deployment Frequency describes how often an item of value is provisioned into the production environment.

High delivery frequency enables companies to respond to market opportunities with the highest-value solutions within the shortest sustainable lead times, thereby outperforming direct competition. In addition, receiving feedback quickly on small, incremental value deliveries enables continuous validation of business hypotheses and the transparency to pivot quickly if a change in direction is required, saving time and costs.

To support release-on-demand capabilities of a business, features must be developed, verified, and stored in a production ready state ready for the organization to release them to the customers. Early, built-in quality measures avoid late, costly delays caused by problem resolution or late refactoring.

## Mean Time To Recover / Mean Time To Repair (MTTR)

To gather meaningful metrics for the MTTR of Off-The-Shelf products, the two dimensions of recover and repair not only need to be considered individually but also combined as necessary:

- Time To Recover is the time it takes to recover a defective product or service for the customer without modifying the product or service.
  - Example SaaS: Infrastructure failure fixed; workaround found
  - Example OTS: Workaround found and verified by customer
  - Example OTS: Customer supported successfully with an integration question
- Time To Repair is the time it takes to fix a broken product or service with a new delivery after a defect was found. Essentially, Time to Repair is identical to Overall Lead Time with a specific filter for critical issues only. Different cases apply:
  - A recovery needs to be followed up with a proper fix (recover acceptable temporarily)
  - A recovery can only be provided with a (defect) fix

The difference between Time To Recover and Time To Repair is that in the latter case, a delivered item of value must be pulled back into the feature development stage of the pipeline and a new delivery has to be made to fix the problem.

In embedded software, Time To Repair is far more common as software fixes/updates are required to resolve issues. These software updates are often delivered via an update during a service interval or via over-the-air (OTA).

## Change Fail Rate (CFR)

Some changes to production or releases to end users result in degradation of the product or service and subsequently require remediation i.e., a hotfix, rollback, fix forward, or patch. The Change Fail Rate is a metric that relates the number of failed changes to the total number of changes:

CFR = Number of failed changes / Number of changes

The focus of the metric is to cover critical defects that impact the end user e.g., product not usable, bad end-user rating, or less turnover. However, the metric does not apply to cosmetic issues with little customer disturbance.

An issue classification model could be used to categorize the severity of issues as only severe issues are considered valid. There may also be severe issues that are raised by the customer but are not valid, e.g., use of an incorrect configuration or simply misuse of the software. The product responsible has to sort out issues that are not valid based on the business model.

As described further above, SaaS software is typically deployed, whereas OTS is typically delivered. Therefore, we propose a refinement for the aforementioned formula below:

### Software as a Service (SaaS)

In a SaaS context, the number of failed changes equals the number of incidents related to a change in a given period of time that either result in degraded service or subsequently require remediation e.g., lead to service impairment or outage, require a hotfix, a rollback, a fix-forward, or a patch. Changes on the other hand are modifications to production including e.g., software eleases, and infrastructure configuration changes.

One possible approximation to a Change Fail Rate in SaaS context is:

CFR = Number of failed changes / Number of deployments

Whereas the number of deployments refers to Deployment Frequency.

### Off-the-shelf (OTS)

In OTS software development, a failed delivery is a software version that has a major impact on the end user e.g., the software is not usable. Here, the Change Fail is determined as:

CFR = Number of failed deliveries / number of deliveries

Whereas the number of deliveries refers to Delivery Frequency.

If a change is a failure or fail is highly dependent on the project context and needs to be defined by the product responsible.

Examples for failures or fails (depends on the concrete project) are:

- Recall campaign
- Infrastructure dependency missed
- Product not useable by customer

## Guidelines For Working with Software Delivery KPIs

Principles [6] to promote transparency and continuous improvement of the end-to-end value chain are:

- Link KPIs to business goals: When developing business goals, management must involve teams in setting those goals and select KPIs that align with and measure progress toward those goals. Focus on outcomes, not outputs.
- Track trends, not numbers: A single data point is not significant, the trend is. Trends show how process changes affect progress toward business goals. Make sure teams are not pitted against each other by comparing team metrics.
- Establish shorter measurement periods: By breaking measurement periods into smaller time periods, teams can review KPIs and trend lines to determine how well they are progressing and identify opportunities for improvement.
- Stop using KPIs that do not drive change: Management and teams need to work on KPIs that drive progress toward business goals and provide verifiable, consistent indicators of progress. If a KPI does not provide a valid indicator, there is only waste and no value in pursuing it further.

### How to improve KPI trends

Usually, software organizations have delivery pipelines. Those pipelines often contain manually performed, slow process steps with significant delays (e.g., handoffs between teams), and require lengthy and error-prone human intervention. This in turn, leads to delaying deliveries, therein increasing their size and scope to avoid integration pain. This approach is the opposite of Lean, which promotes limiting work in progress (WIP) and reducing batch size to optimize Flow and increase the delivery frequency.

While KPIs provide transparency of the overall flow of value and delivery performance, they do not always have clear-cut answers or actionable tips that will seal an organization's success. To improve their performance, teams need more granular metrics. These metrics should enable deriving concrete actions to improve value flow and drive the overarching paradigm of frequent deliveries or deployments at high quality.

In best case, those granular metrics continuously monitor the state of the delivery pipeline. The first step in improving value flow is to map the current pipeline [7] to understand delays and identify opportunities for improvement, such as eliminating delays or reducing rework. When systems are first mapped, it is common to not have clear metrics for all steps immediately. Filling in those gaps is already a valuable finding and improvement. The second step is to continuously monitor this mapping through appropriate metrics.

Four granular metrics are used for providing transparency on KPIs along the mapped value flow:

1. Lead time for each process is the sum of Delay Time from the completion of the previous step and Process Time of the current step. The lead time of each step in the pipeline adds up to the Overall Lead Time of value flowing through the delivery pipeline.
2. Process Time is the time required to complete the work in one step of the delivery pipeline. Very long Process Time(s) can be an indicator for work not broken out modularly enough, i.e., sufficiently small work items. If this is the case, a correlation with poor Percent Complete and Accurate (%C&A) and overall poor Delivery Frequency is likely as work may be planned too monolithically.
3. Delay Time is the time when no work is taking place. An example of this is when work to be accepted by the Product Manager is significantly delayed because he/she is a bottleneck or not available. Understanding and eliminating unnecessary delays is critical to improving value flow.
4. Percent Complete and Accurate (%C&A) represents the percentage of work that the next step can process without rework. One can think of it as the reverse of Fail Rate for a particular process step. Often, delays are caused by poor quality in upstream steps. The %C&A metric helps identify the steps where poor quality occurs and causes longer lead times, resulting in delays in value delivery and in the cases of defects a longer Time to Repair. Improving the %C&A metric is also essential to improving value flow and therefore Overall Lead Time or Time to Repair.

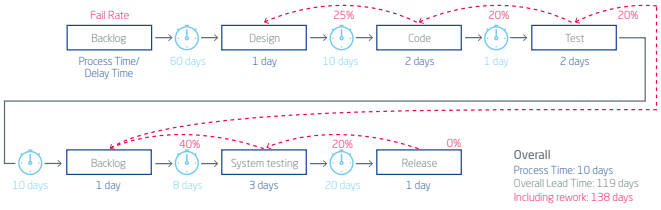


Fig. 3: Effect on Overall Lead Time in case of not accurate-and-complete over multiple steps on the process pipeline (fictional example)

Taken together, these granular metrics provide a detailed breakdown for velocity and quality at each step in the delivery pipeline.

Detailed monitoring of the value flow shows which measures are best used to reduce the Overall Lead Time and Mean Time To Repair (MTTR), as well as to increase Deployment Frequency / Delivery Frequency. Often, the most important factor is Delay

Time. Reducing Delay Time is usually the most effective way to reduce Overall Lead Time and/or increase Delivery Frequency. Another high-priority area for improvement is any step with low %C&A metrics, as reducing rework allows teams to focus on adding value to the product or service. Subsequent improvement opportunities typically focus on reducing batch size to improve frequency of delivery / deployment. Over time, teams continuously work to improve the efficiency of each step and monitor the impact of these improvements based on the trend shown in the software delivery performance KPIs, i.e. Overall Lead Time, Delivery Frequency, Mean Time to Repair, Change Fail Rate.

## ETAS Use Case

The engineering organization of ETAS thrives to be among the leaders in the industry with regards to development and automation practices. ETAS operates in an agile culture following the Scaled Agile Framework (SAFe) and applies a continuous improvement mentality [8].

Software engineering occurs organized in Agile Release Trains and is planned in quarterly Program Increments. In preparation of the Program Increment planning sessions, Product Managers & Product Owners prioritize and refine the Solution Backlog. Since adopting Scrum almost a decade ago, the ETAS development teams have continuously improved their delivery pipeline to be able to deliver incremental value weekly or daily depending on the product. Based on market demand and the ability of our customers to consume releases, ETAS releases new versions to the market on a fixed cadence of 3 months for most products.

Following the publication of the book »Accelerate« by Nicole Forsgren, Jez Humble, and Gene Kim in 2018, ETAS decided to implement the KPIs Overall Lead Time, Delivery Frequency, and Mean Time to Restore/Repair (MTTR). To drive this to completion, a dedicated DevOps champion was hired to the engineering organization in late 2018 and a dedicated CI/CD Chapter was established to develop patterns on how to measure and leverage these KPIs as well as expedite the implementation of CI/CD methodology in the various product development organizations. Despite initial skepticism, the CI/CD Chapter succeeded in convincing the development teams of the benefits of transparency on these three KPIs. The fourth KPI discussed in the book, Change Fail Rate, was put on the backlog to be applied to future Software as a Service (SaaS) solutions because it is often difficult to identify the point in time of the injection of a problem for Off-the-Shelf products. In many cases, defects are found years later by changing usage patterns by customers.

Currently, Overall Lead Time in ETAS is measured starting from the point in time when a Program Manager pulls an unqualified item from a pool of ideas to the backlog. The measurement ends at the point in time when all engineering tasks are completed and verified leaving only release specific – mostly business and legal related – activities to be done.

For ETAS' Off-the-Shelf (OTS) software products, this is typically the step when the binaries are placed in an artifact repository or onto a download server. For complex systems including hardware, some long running and expensive final validation steps are performed on demand only whenever a release is planned. Technically, validation steps have to be performed before the product can be delivered. As some long-running and expensive final validation steps are more closely linked to the release decision, they are excluded from the measurement of Overall Lead Time.

As can be seen in the example in Fig. 4, measurement of Overall Lead Time is comprised of the lead times of the process steps involved has made transparent that in many cases the time consumed in the exploration or preparation phases of any new feature is significantly higher than the time required for implementation and verification. Such insight has helped to shift the focus to the appropriate steps in the value chain to analyze potential for efficiency gains leading to a steady reduction in the Overall Lead Time.

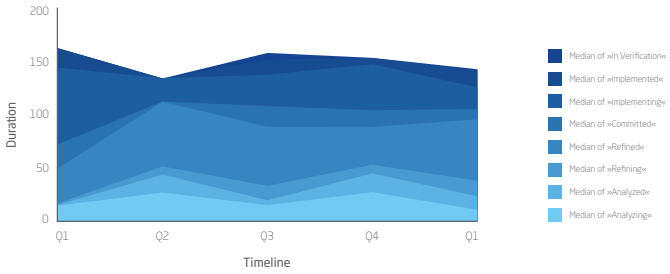


Fig. 4: Example of Overall Lead Time distribution per process step for an OTS software product

By measuring Mean Time to Repair, a distinction has been made transparent between the time engineering needs to develop a fix and the total time needed until the defect is resolved at the customer site.

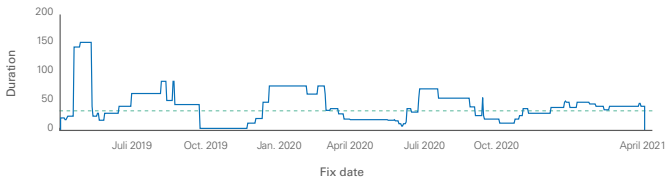


Fig. 5: Evolving Mean Time to Repair trend for an OTS software product

Due to the fact that ETAS does not have a channel to directly deploy a software change into a customer installation, any fix is bundled up with the next scheduled quarterly release. This results in a time window where every fix waits for the next release. This insight has stimulated a lively discussion to address the tension that ETAS' customers simultaneously want fixes delivered as quickly as possible on the one hand and often cannot consume a higher Delivery Frequency on the other.

Through investments in the automation of testing and artifact & document creation, some ETAS teams were able to increase delivery frequency from 2-3 deliveries per week in 2019 to

greater than 1 delivery per day by Q1 2021. This investment significantly reduces the transaction cost of process steps during the end phase of the delivery pipeline and enables the teams to support a growing market with increasing demand for variations. Based on the success observed the team plans to further increase delivery frequency in 2021.

## Summary

In an increasingly volatile environment, the time it takes to validate a business hypothesis or fulfill a customer requirement, combined with the ability to deliver at optimal frequency is becoming increasingly business critical. Building and maintaining a delivery pipeline gives an organization the ability to continuously deliver new high-quality features to customers faster and far more frequently.

In this sense, this whitepaper refers to the four KPIs identified by Forsgren et al. (2018), which are causally related to improved business performance [1] and proposes that for businesses selling software products such as Off-the-shelf (OTS) and Software-as-a-Service (SaaS), an adapted set of KPIs is used. In this context, we conclude that Overall Lead Time, Delivery respective Deployment Frequency, Mean Time To Restore/Repair, and Change Fail Rate have proven useful as KPIs.

These four KPIs provide software organizations with an effective tool for continuous learning and improvement and allow teams and organizations to compare and benchmark against other teams within the organization or against the industry as a whole.

Although measurables do not always have clear-cut answers or actionable tips that will seal business success, the benefit of well-chosen KPIs is that they foster regular discussions and nourish a culture of continuous improvement.

## References

[1] Forsgren, Nicole; Humble, Jez; Kim, Gene (2018): Accelerate: The Science of Lean Software and Devops: Building and Scaling High Performing Technology Organizations.

[2] Marr, Bernard (2020): What's The Difference Between Lagging And Leading Indicator?
Link: https://www.forbes.com/sites/bernard-marr/2020/10/23/whats-the-differencebetween- lagging-and-leading-indicator

[3] Hatheway, Richard (2016): The Real Difference Between Metrics and KPIs.
Link: https://www.linkedin.com/pulse/real-difference-between-metrics-kpis-richard-hatheway

[4] Weber, Sergej; Tengler, Steve (2020): Five Ways Agile in Automotive Will Pivot in 2020.
Link: https://www.wardsauto.com/industry-voices/five-ways-agile-automotive-will-pivot-2020

[5] Thoughtworks (2018): Four Key Metrics.
Link: https://www.thoughtworks.com/radar/techniques/four-key-metrics

[6] Altvater, Alexandra (2017): What Are Software Metrics and How Can You Track Them?
Link: https://stackify.com/tracksoftware-metrics

[7] Scaled Agile, Inc. (2021): Continuous Delivery Pipeline.
Link: https://www.scaledagileframework.com/continuousdelivery-pipeline

[8] Scaled Agile, Inc. (2021): Scaled Agile Framework 5.1.
Link: https://www.scaledagileframework.com

[9] ETAS (2021): About ETAS.
Link: https://www.etas.com/en/company/about-etas.php

[10] Kugler Maag Cie (2021): About Us.
Link: https://www.kuglermaag.com/about-us