POLITECNICO DI TORINO

Master's Degree in Engineering and Management
A.a. 2023/2024
Sessione di Laurea Marzo/ Aprile 2024

Tailored implementation of SAFe and Scrum in a Real-Word Project:
A case study

Relatore:
Marco Cantamessa

Candidato:
Davide Angelo Laudati

Summary

Abstract

This master's thesis explores the transition from traditional waterfall project management methodologies to Agile frameworks, specifically focusing on the Scaled Agile Framework (SAFe) and Scrum. The study begins with a focus on the traditional waterfall approach, and some frameworks with which it is applied. It delves into the key principles of SAFe and Scrum, highlighting their differences from traditional approaches and emphasizing their benefits in fostering adaptability, collaboration, and iterative development.

A real-world project serves as the focal point of the research, providing a practical context for understanding the implementation of SAFe and Scrum methodologies. The initial section of the thesis sets the stage by explaining the project environment and delineating its scope. This includes factors such as organizational structure, team composition and project objectives.

The subsequent portion of the thesis delves into the project's management using the SAFe methodology. It examines how SAFe principles are applied to facilitate alignment, synchronization, and transparency across multiple teams, ensuring the efficient delivery of value to customers.

In the final phase of the study, the focus shifts to the adoption of Scrum processes within the project framework. An analysis of the core Scrum practices, including backlog refinement, capacity planning, estimations, and retrospectives, is conducted to evaluate their efficacy in driving iterative development and mitigating risks. Additionally, the thesis examines the challenges encountered during the implementation of Scrum practices and the strategies employed to address these challenges, thereby enhancing the overall agility and effectiveness of the project.

By integrating theoretical insights with empirical observations from a real-world project, this thesis provides valuable insights into the practical implications of apply the Agile methodologies, specifically within the context of SAFe and Scrum. The findings contribute to a deeper understanding of the complexities involved in agile project management and offer practical recommendations for Reply companies seeking to increasingly embrace Agile methodology in their projects.

Introduction

In today's rapidly evolving business landscape, effective project management methodologies are essential for organizations to achieve success in their endeavors. This thesis explores various project management methodologies, with a particular focus on traditional Waterfall and Agile approaches, and their applications within modern project environments. By examining these methodologies, their frameworks, and their integration into organizational processes, this study aims to provide insights into optimizing project management practices for improved efficiency and outcomes.

The first chapter introduces the fundamental concepts of project management and discusses different methodologies utilized in project execution. It begins with an exploration of the Waterfall methodology, detailing its sequential approach to project phases. Subsequently, it delves into specific Waterfall frameworks such as the Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT), highlighting their significance in project scheduling and management.

Going on, chapter 2 shifts focus on Agile methodology, emphasizing its adaptive and iterative nature in contrast to the rigid structure of Waterfall. Within this context, the chapter examines the Scaled Agile Framework (SAFe), underlining its core values, the value stream proposed by the method, and Agile Release Train (ART) structure. Finally, it discusses the principles of Scrum, a popular Agile framework, including its processes, team dynamics, artifacts, and ceremonies.

The third chapter provides an overview of the Reply company and a detailed examination of the project's context. It traces the historical background of the organization, offering insights into its evolution and strategic direction. Furthermore, it outlines the scope of the project, the teams involved, and the overarching objectives guiding the project execution.

Chapter 4 details the project plan, encompassing its first phases, processes, and timeline. It delineates the initial phase of the project and provides a high-level overview of the processes involved. Additionally, it presents a comprehensive timeline outlining key milestones and deliverables throughout the project lifecycle.

Moving forward, the fifth chapter is the core of the thesis. It explores the integration of Agile processes into the project framework, focusing on backlog refinement, capacity planning, key performance indicators (KPIs), risk management, and retrospective analysis. It is not only a description of the main processes adopted, but through a deeply literature research, the thesis provides improvements adopted and measured over the past year and a half on the project.

In the final chapter, the results of the study are summarised, highlighting the benefits and implications of adopting Agile methodologies in the context of project management. The importance of flexible and adaptive approaches in meeting evolving project requirements and achieving successful outcomes is emphasised.

Through an in-depth examination of project management methodologies and their practical application within organisational contexts, this thesis aims to contribute to the body of knowledge surrounding effective project management practices in today's dynamic business environment and as a bridge between the theory of methodologies and the practical implementation of them.

1 Project Management and its methodologies

The objective of this chapter is to explore the evolution of project management, from its inception to its current widespread adoption. Furthermore, it aims to undertake a comparative examination of two major methodological frameworks: the Waterfall and Agile models, specifically on the two methodologies SAFe and Scrum.

1.1 Waterfall methodology

The first methodology adopted in software development was the Waterfall model, introduced by Winston Royce in 1970. This approach consists of sequential steps [1] that describe each essential phase of the project and are described in the following section.

Requirements

The most important aspect of this phase is to understand what the customer wants. Indeed, it's needed to work together with the client for building up a list of requirements called "Software Requirement Specification". It's important to underline that in this phase it's described the "what" of the system and not the "how". The requirements must be clear before starting the next phase, and cannot be changed during the other phases; this is a key aspect of this methodology, in this way all the other phases of the development of project can be planned without taking again into.

Analysis Phase

During this step, the system specifications are examined to generate models and logic that guide the creation of the product. This is also the stage where technical and financial resources are assessed for their feasibility.

Design Phase

In this phase, a detailed design specification is formulated, outlining technical requirements such as programming languages, hardware, architecture, data sources, and services.

Coding and Implementation Phase

Here, the actual source code is developed based on the models and specifications created in the earlier phases. The coding often occurs in smaller components or units before being integrated into the final system.

Testing Phase

Quality assurance, unit testing, system testing, and beta testing are carried out in this phase to identify and rectify any issues. After this a phase of testing in end-to-end environment is carried out, it is different from the previous one because this type of testing approach starts from the end's user perspective and simulates a real-world scenario. If problems are encountered, the coding phase may need to be revised for debugging.

Operation and Deployment Phase

At this point, the product or application is considered fully functional and it is deployed in the production environment where users can access it.

Maintenance Phase

This phase involves continuous maintenance activities, including corrective, adaptive and improvement activities. It encompasses tasks such as releasing patches and updates. Despite the massive phase of testing, most of the time some bugs are discovered only once the product was released into the production environment, it's really complicate to cover all the corner case during the test fase. Accordingly, this life-cycle restarted from point 3, in order to deployed a new version of the product with the corrective solution.

1.2 Waterfall Frameworks: Critical Path Method

The Critical Path Method (CPM) is an advanced project management technique used to meticulously plan, schedule, and oversee intricate projects [2]. It entails delineating all necessary tasks for project completion, establishing their sequential order, and calculating the longest duration required to finalize the project from inception to conclusion. This longest duration, termed the "critical path", serves as a definitive timeline for project culmination.

In CPM, tasks are classified as either "critical" or "non-critical". Critical tasks are imperative for maintaining the project schedule; any delays in these tasks will invariably delay the project's overall completion. Conversely, non-critical tasks afford some scheduling flexibility and are less likely to impede project timelines.

CPM proves particularly beneficial for projects comprising numerous interdependent activities, as it enables project managers to highlight potential bottlenecks and allocate resources optimally. By discerning the critical path, project managers can prioritize tasks, mitigate risks, and enhance the likelihood of timely project fulfilment.

There are several benefits of adopting this framework. First of all, a clear visualization of the project timeline is provided. By outlining task sequence and duration, CPM offers a comprehensive view of the project's flow, facilitating resource planning and management. Going forward with the analysis, is it possible to identify the critical tasks which helps in task prioritization and proactive management of potential delays, enabling better adaptation to changes or issues during the project lifecycle. Highlighting the critical path allows for early detection of potential bottlenecks or delays, enabling proactive risk mitigation and reducing project downtime and associated costs.

In summary, CPM serves as an efficient tool in project management, offering enhanced planning, risk mitigation, resource management, and team collaboration capabilities.

1.3 Waterfall Frameworks: Program Evaluation and Review Technique (PERT)

PERT was introduced to enhance the Program Manager's control in projects where time was of crucial importance and time estimates were challenging to determine confidently. This technique is event-oriented, focusing on the start and finish of activities [3]. PERT utilizes three time estimates for each activity: optimistic, pessimistic, and most likely. Expected time is then calculated based on a beta probability distribution derived from these estimates.

The six steps described below gives a clear overview on the PERT framework:

1. Identify Activities: The initial phase involves identifying the necessary steps to complete the project. Each task should be specific, measurable, and assigned a predetermined duration. For instance, in a creation of a startup, the main phases are conduct competitive research, establish a team, outline core features, create mockups, coding etc..

2. Figure out dependencies: highlighting potential dependencies among the tasks, in this way it will be a clear overview on which tasks could be done in order, while other simultaneously. This step assists in understanding the

interdependencies among various tasks and their impact on the project timeline.

3. Estimate duration: for each task an estimation should be provided. This should be done by specific experts into the related field. It is an essential step to figure out the whole project schedule.

4. Make a network diagram: gather all the information discovered in the previous step and make a network diagram that demonstrate how the tasks are related to each other. This diagram typically employs nodes to represent actions and arrows to depict the sequence and progression.

5. Find the critical path: Identify the critical path, which represents the longest sequence of interdependent tasks which shows the project's overall duration.

6. Schedule and Control: Develop a schedule based on the expected times derived from critical path analysis. This schedule outlines the project's execution plan and facilitates tracking and management of progress. Regular monitoring, tracking, and adjustments are essential to ensure project adherence and prompt resolution of any issues that arise.

Although the Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT) share similarities, there are significant differences that make one technique better suited for certain projects than the other. PERT excels in situations characterized by significant uncertainty and where time management takes precedence over cost control. It addresses time uncertainty by generating three estimates for each activity and then calculating an expected duration using the beta distribution. In contrast, CPM is more suitable for projects with well-defined parameters and minimal uncertainty, allowing for accurate estimates of time and resources. Additionally, CPM enables the determination of activity completion percentages.

2 Agile Methodology

The origins of Agile methodology trace back to the late 1990s and early 2000s when traditional software development practices were under investigation for their effectiveness [4]. In response to these challenges, a group of influential practitioners gathered in Oregon in the spring of 2000. Their objective was to expedite development timelines, improve software delivery, and prioritize customer satisfaction.

During this time, two critical opportunities were identified: the need to accelerate the delivery of benefits to users and to gather prompt feedback for continuous improvement. This marked a paradigm shift, emphasizing adaptability, collaboration, and customer-centricity.

The evolution of Agile methodology represents a significant transformation in software development and project management. It has redefined organizational success by embracing change, emphasizing human interactions, and focusing on delivering value to customers.

Two of the most famous frameworks used in Agile are SAFe and Scrum, which are described in the following paragraph.

2.1 SAFe

As the name suggests, Scaled Agile Framework is a framework for the implementation of the Agile Project Management methodology on a large scale in big companies. Specifically, as explained by the inventor Dean Leffingwell, it's a set of proven patterns on how to apply lean agile development at enterprise scale. Being designed for a rapidly changing industry, SAFe has grown with the market because new and better ways of developing software and systems have been discovered since 2011. Therefore, it passed from the version 1.0 to nowadays, version 6.0 [5] that has been released in March 2023.

SAFe, a relatively recent Agile Project Management methodology, leverages the collective knowledge of established development methods and processes. Its primary objective is to provide substantial advantages to businesses that embrace this framework by integrating principles from Agile, Scrum, Extreme Programming, and Lean. SAFe also relies on the principles of product development flow, which teams can employ to create and deliver functional software.

As Agile methods have gained traction within the corporate landscape, several extensions to the fundamental agile practices have become essential. These extensions address broader challenges related to process, organization, scope, and governance commonly encountered in larger enterprises. Indeed, one of these challenges involves adapting agile team practices (such as product backlogs and user stories) to the requirements of program and portfolio levels within the company. Vertical Scaling: several independent teams jointly planning, developing, integrating, testing and deploying a product based on an integrated product vision.

The primary motivation behind developing this methodology is to facilitate the organization and management of agile practices within large enterprises. To achieve this objective, the framework is divided into three distinct perspectives: team level, program level, and portfolio level. The boundaries between these three levels are arbitrarily defined and serve as a model for abstracting the scope and scale across these levels.

Scaling agile is an excellent way to boost productivity and drive innovation.

2.1.1 Benefits of Scaling Agile

Scaling agile is an excellent way to boost productivity and drive innovation. The main benefits are listed below:

1. Offers enterprise-wide visibility, strategic alignment, and planning: it makes easier for the teams to work together for delivering strategic objectives and a wide visibility offers the opportunity to find and react in a faster way to possible impediments.

2. It guarantees heightened productivity, so scaling agile through the entire organizations brings all departments under the same task management. This leads to a transparent workflow, making immediately visible who is accountable for what actions in this way the downtime between tasks will be reduced.

3. Task based decisions: reducing the effort needed by managers for managing projects and empowering the team members to make more work-related decisions;

4. Program Increment (PI) planning: opportunity for teams to build a product plans, following the common vision. Indeed, through this meeting people from multiple departments can coordinate and they will have a clear visibility on the status of the other teams and on what they achieved.

2.2.2 Core Values SAFe

The SAFe core values [6] demonstrate the behaviour and action for everyone who participate in the SAFe portfolio.

The core values are four:

1. Alignment: in an environment undergoing rapid change being aligned is crucial because it allows the company to react quickly. Alignment means that each team can understand the current state of the business, the goals and how everyone should move together to achieve those goals.

2. Built-in-quality: each part of the solution should reflect the quality standards. According to SAFe there are five key dimensions of built-in quality: flow architecture and design quality, code quality, system quality and release quality.

3. Transparency: points of failure are always present; therefore, openness is very important. To ensure openness trust is the key encouraging trust building behaviour including planning work in smaller batch sizes so problems can be surfaced and solved sooner.

4. Program execution: the teams have to execute and continuously deliver quality working software and business value on a regular basis.

2.2.3 Value Stream

SAFe's definition of the value stream is a series of steps that an organization uses to implement Solutions that provide for the continuous flow of value to the customer. Solutions is meant products, services, or systems delivered to the end customer either internally or outside to the company.

As it is possible to visualize in the figure below, the value stream started with a trigger that is usually the request for a product or service and it ended with the delivery of a product or service at right time.

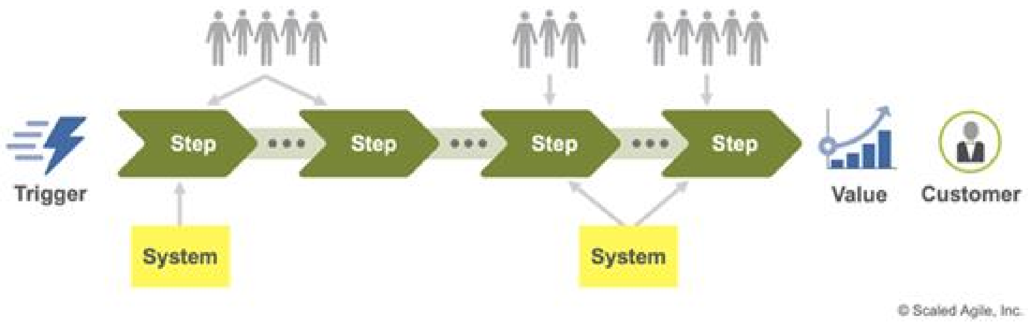


Figure 1: SAFe Value Stream (https://scaledagileframework.com)

To make this process to happen there are a series of steps in between that must occur and the sum of time that each of these steps, it represents the total lead time.

SAFe framework recognizes two types of Value Stream [7]:

- Development Value Stream (DVS): describes the process and individuals responsible for creating solutions utilized within operational value streams.

- Operational Value Stream (OVS): encompasses the processes and individuals involved in providing end-user value through solutions generated by development value streams.

The main benefit of organizing SAFe around a Value Stream is to improve workflow, efficiency and accelerate time to market. The other benefits are listed below:

- Foster the formation of enduring, reliable teams with a dedicated focus on value delivery.

- Facilitate the identification and clear visualization of all the tasks required to develop solutions.

- Promote transparency in revealing delays, bottlenecks, and handovers in the process.

- Encourage the adoption of smaller work batches for increased efficiency.

- Cultivate knowledge expansion and a continuous learning environment.

- Enable a transition in the funding model, moving away from traditional project budgets towards value stream-based financing.

2.2.4 Agile Release Train

Agile Release Train (ART) refers to long-lived group of Agile teams, who along with other stakeholders and incrementally plan, commit, develop, and deploy one or more solutions that deliver benefit to the end user [8].

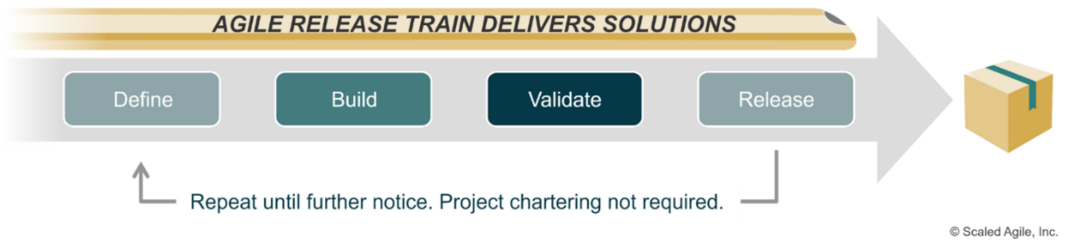

AGILE RELEASE TRAIN DELIVERS SOLUTIONS
Define
Build
Validate
Release
Repeat until further notice. Project chartering not required.
© Scaled Agile, Inc.

Figure 2: SAFe Agile Release Train Delivers Solutions (https://scaledagileframework.com)

Typically, in each release train are present from 5 to 12 agile teams for a total of 50-125 overall people, according to literature.

The ART organization deeply changes conventional functional divisions that may be in place (the segmentation of work across separate departments) and employs systems thinking to construct a cross-functional structure designed to streamline the flow of value from the conception of ideas through to deployment, release, and operational stages.

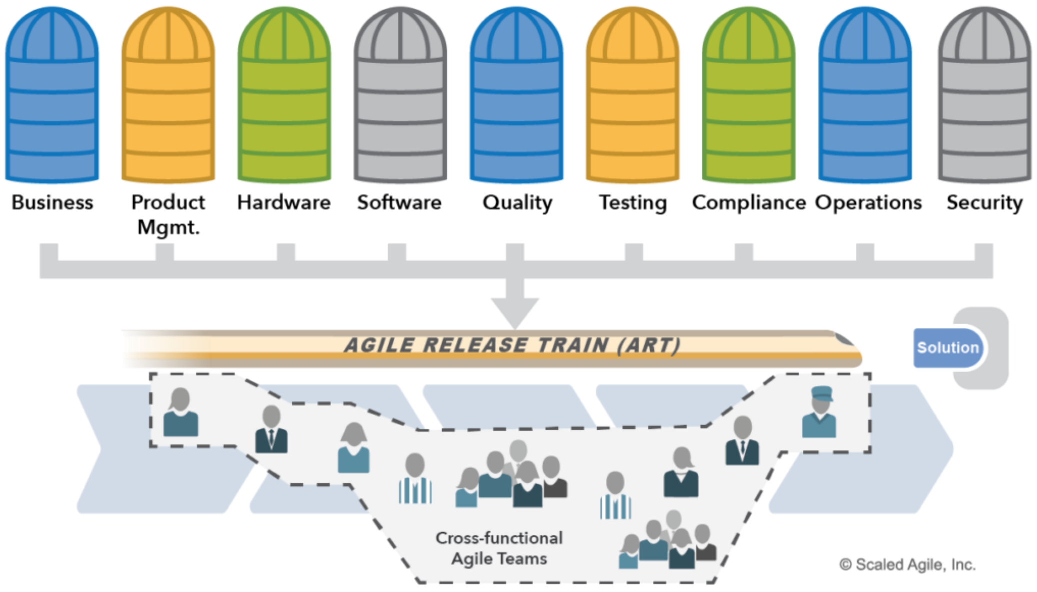


Figure 3: SAFe ART (https://seibert.group)

Agile teams power the Agile Release Train. Each teams have 5 to 11 individuals with cross-functional capabilities, which are able to build a quality increment of value every iteration. Then, individually the team can choose an agile framework for setting the work inside the team. The frameworks that are usually used are Scrum, XP, and Kanban and each agile team has two speciality roles, the Scrum Master and the Product Owner.

2.2.5 ART Responsibilities

Into the Agile Release Train it is possible to identify 5 crucial responsibilities as depicted in the image below [8].

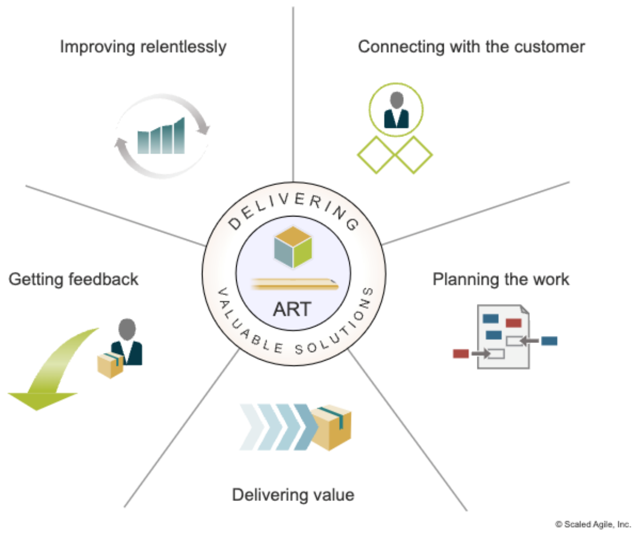


Figure 4: SAFe ART Responsabilities (https://seibert.group)

Connecting with the customers

The main beneficiaries of business solutions implemented and supported by Agile Release Trains (ART) are the customers.. However, establishing a meaningful connection with customers necessitates purposeful endeavours and a comprehensive understanding of how to implement lean and agile principles within the distinct context of an ART. To achieve this, it is necessary to apply customer centricity, that is, to focus daily on the customer's needs and opportunities to benefit from them. This will create a benefit in the relationship established with the customer and maintain an empathy with the customer and a continuous search for the best applicable solution. The other point is the use of design thinking, which is a recurring process of understanding and discovering possible problems and designing the right solution. The use of lightweight prototypes quickly validates customer value assumptions and keeps ART on the right track.

Planning the work

It is a crucial activity inside the ART because enables teams and stakeholders to share and understand what and how to build in the next period. All the main players involved in the PI planning must arrive prepare for the event, teams take inventory of their remaining work in order to estimate the available capacity for the PI and try to discover any possible risks that can arise in the future. In addition, teams create and agree on the PI Objectives that will guide them throughout the PI. Product Management and Business Owners develop the vision and agree on priorities for the next PI. Another key aspect of this event it is to create alignment within the ART, indeed Business Owners can share business and customer context with teams, on the contrary they can learn from the team more on the current technology and the delivery capability.

Delivering value

It's important to establish a cadence in which the ART will release the value created to the customers. It is necessary to break down the PI as a kind of small increments, each of which should reflect a small part of integrable, tested, and demonstrable value. Having a process that quickly manages to integrate all phases of development such as design, implementation, testing, and release allows for a learning curve for all teams involved in the project, which is extremely fast because you will have constant feedback through the entire duration of the project. The ultimate goal of SAFe is to build a Continuous Delivery Pipeline (CDP). This is a complex process, as it has two main critical issues, the first being that it requires a complete and thorough overview of the entire value stream, so as to know exactly the sources of excessive delay and variability. In addition, it is necessary to have a system design that helps low coupling of capabilities, which enables the teams to deploy value independent of each other.

Getting feedback

To have a highly efficient ART a highly development velocity is needed and most of the time speed comes from fast learning and adaption. SAFe framework proposes a routine to be implemented for achieving this:

- Involving customers in the development process allows you to be as close as possible to the customer's desired solution and avoid the cost and time of rework.

- Once the final solution is released to the client, it is important that it is measured business outcomes and usage. It is relevant because customer usage can detect problems and opportunities, which otherwise would have remained invisible to ART.

- After all the tests performed by the team it's important to test from the client perspective. Indeed, usually there is an end-to-end testing environment, in which the testers can verify the entire software application, from start to finish, including all the systems, components, and integrations involved in the application's workflow. This is done before the last release, there will be in production, i.e, available for use by the end customer. For this reason, is important to ensure that the application works correctly and perfectly match the user requirements.

Improving relentlessly

Do not be stuck! An ART seeks to continuously improve productivity in delivering customer value. For achieving a process in which the improvement is continuous, it is needed to measure different aspects of an ART and identify areas of improvement. Indeed, at the end of each PI there is the opportunity to look back, identify problems and take corrective actions. Division into PIs, sets a perfect cadence so that ART can continuously and steadily improve.

2.2.6 Crucial ART Roles

The crucial roles within ART are listed below, and it is essential to understand who is responsible for what in order to determine the main actors involved [8].

Release Train Engineer (RTE): is a servant leader and a coach for the Agile Release Train. The RTE is a facilitator of the agile release train events and processes and assist the teams in any way possible for delivering value. The other main tasks of an RTE: owner and reporter of impediments if the situation required an escalation, maintains relationships and communicates with stakeholders, helps for managing risks and drives relentless continuous improvement.

Product Management is responsible for defining and supporting the building of a product as defined but the vision roadmap and new features in the program backlog. For doing this, they have to collaborate with a wide range of people as customers, product managers, product owners, and suppliers. This is necessary for identifying and defining customer needs, developing program vision, and roadmap and features required to meet these needs.

System Architect is an individual or a team who is responsible for defining and communicating the overall architecture of the system for an ART, aligning all the teams on the technical direction to follow. To do this requires ongoing analytical work that will lead to possible technical trade-offs, identifying primary components and subsystems.

Business Owners are key stakeholders and play a critical role in helping the ART in delivering value to the Customers, which are playing another crucial role. Indeed, they have ultimate responsibility for the business outcomes of the train. Customers are the ultimate economic buyer or value users of the solution.

2.3 Scrum

Agile scrum methodology is a project management system that relies on incremental development. Scrum derived from “scrummage”, a term that comes from rugby where it means to restart play after there’s been a minor rule infringement. During a scrum, players huddle closely to compete for ball possession and collaborate to advance it down the field, this is exactly what you want to apply to the IT world, all team members working closely together to develop and deliver value. This analogy was introduced by Hirotaka Takeuchi and Ikujiro Nonaka in a 1986 Harvard Business Review paper called “ The New Product Development Game” which is referring to company as Honda, Canon and Fuji-Xerox.

2.3.1 Scrum Framework

Scrum is a lightweight framework primarily composed of iterative practices. As it is shown, in the figure 5, the Product Backlog is generated based on inputs from stakeholders, outlining a set of actions. Subsequently, the planning of tasks to be accomplished in the upcoming sprint is detailed and documented in the Sprint Backlog. Throughout the sprint, the objective is to successfully complete all tasks. The team remains continuously informed through daily meetings, addressing any issues that may arise autonomously.

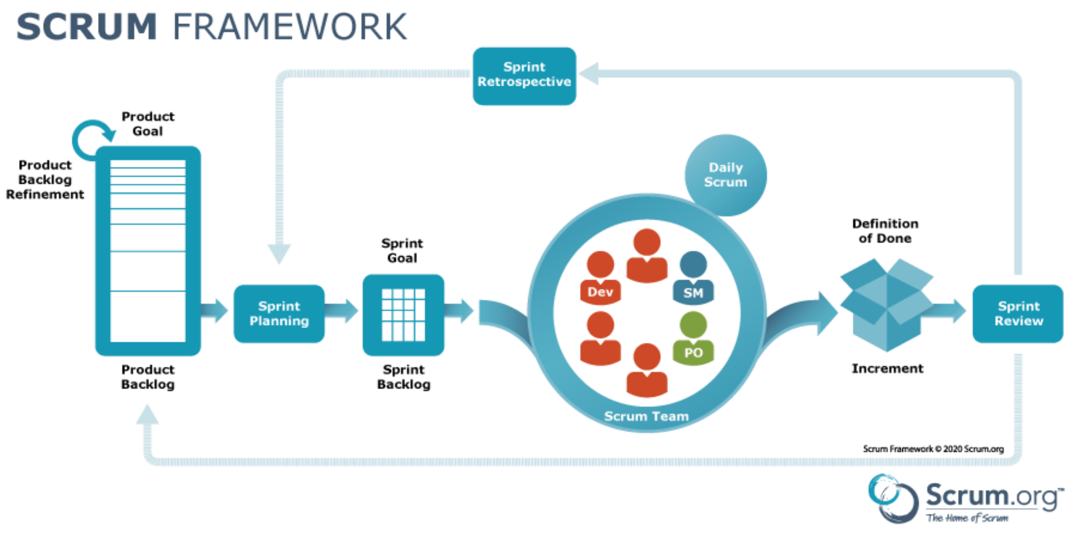


Figure 5: Scrum Framework
(https://www.scrum.org/resources/scrum-framework-poster)

At the conclusion of the sprint, two scheduled meetings take place: the Review, where stakeholders are briefed on the increment in the product's value that has been released, and the Retrospective, where team members discuss the ongoing process. Following this, items in the Product Backlog undergo review, modification, and the addition of new ones. A new sprint then commences, and the cycle repeats. Each component of the Scrum framework will be thoroughly explained in the upcoming sections [9].

2.3.2 Scrum Team

A Scrum Team consists of a group of people, usually ranging from five to nine members, collaborating to deliver the necessary product increments. The scrum framework encourages the team to work in the same location for having a smoother communication among team members, this can easier bring the team to follow a common goal, adhere the same norms and rules and show respect to each other [10]. The Scrum team, as it is possible to see in the figure 6, includes, the Product Owner, the Scrum Master and the Development Team.

The main characteristic of a scrum team lies in the fact that it is self-organized and cross functional, this makes it so that there is no hierarchy, only a cohesive team working toward the same goal. Wanting to build a team, based on certain characteristics allows one to stimulate creativity and be more flexible, ensures that the team remains increasingly aligned with the agile philosophy.

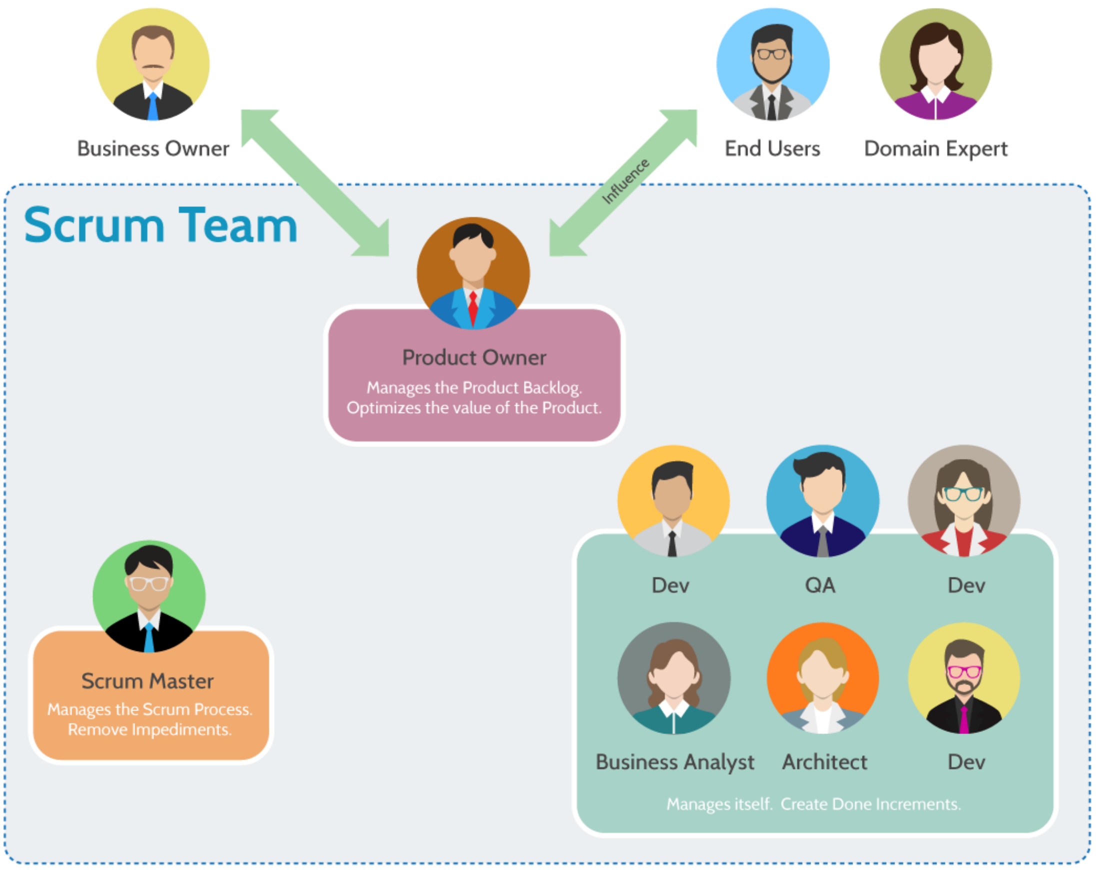


Figure 6: Scrum Team
(https://www.visual-paradigm.com/scrum/what-is-scrum-team/)

Product Owner

The Product Owner plays a crucial role within the Scrum team, as he is the link between customer needs and the team's work. He has a deep understanding of what the customer wants, and the business value associated with those desires. This knowledge enables him to effectively convey the customer's requirements and values to the Scrum team. The Product Owner must also fully engage in the business case for the product and the specific features that customers are seeking.

In addition to their role as a communicator, the Product Owner is responsible for managing the Product Backlog. This involves expressing Product Backlog items clearly, prioritizing them to achieve goals efficiently, and optimizing the value of the team's work. The Product Owner ensures that the Product Backlog is visible, transparent, and comprehensible to all team members, providing clarity on the team's upcoming work. They also ensure that the team has the necessary understanding of the items in the Product Backlog to execute them effectively. Ultimately, the Product Owner has the authority to make critical decisions to complete the project successfully, indeed he is ultimately responsible for delivering the product to the end customer.

Scrum Master

The Scrum Master plays a crucial role in guiding and supporting the Development Team and the Product Owner in their day-to-day development activities. He ensures that the team embraces and practices Scrum values and principles, fostering enthusiasm for Agile and promoting self-organization. The Scrum Master also educates and motivates the team, encourages communication and collaboration, and acts as a process leader to help others understand Scrum's values and principles.

The Scrum Master fulfils multiple roles within the team. Indeed, they coach the Development Team and the Product Owner, fostering effective communication and addressing obstacles. They also serve as a facilitator for the Scrum Team, organizing Scrum Events and assisting in crucial decisions to boost productivity. Furthermore, the Scrum Master is responsible for removing impediments that hinder the team's ability to deliver business value, prioritizing and resolving these issues. They act as a shield, protecting the team from external interference and distractions, in this way the team can only focus on delivering value. In addition, they work constantly close to the team for improving enhancing processes and practices in order to always improve the performance of the team and ensuring the achievement of the sprint goal.

Development Team

The Development Team is a group of professionals that are working together for delivering a valuable and done increment at the end of each sprint.

A Development Team have the following characteristics:

- The Development Team is self-organizing, meaning they have the autonomy to determine how to transform items from the Product Backlog into product increments.

- Development Teams are cross-functional, having all the necessary skills within the team to create a product increment.

- Scrum does not assign specific titles to individual Development Team members, regardless of their roles and responsibilities.

- Scrum does not recognize sub-teams within the Development Team, even when different domains such as testing, architecture, operations, or business analysis need to be addressed. The accountability belongs to the Development Team as a whole.

- An optimal size for a development team should be between 3 and 7, in this way it can be ensured that the development team has all the cross functional skills needed for the project and is not too large to avoid introducing complexity at the level of team coordination and self-organization of the team.

2.3.3 Scrum Artifacts

Scrum artifacts play a crucial role as tools to enhance team efficiency. Ensuring that all the people strictly involved into the project, have access to and visibility of these artifacts is vital for the long-term success of the project. Regular evaluations and discussions of the artifacts by product managers and Scrum Masters with development teams are essential. This ongoing engagement helps teams put the light on operational inefficiencies and fosters the generation of innovative approaches to enhance productivity and efficiency. For all these reasons it’s crucial the transparency of the artifacts, because the decisions aimed at optimizing value and managing risk are made based on the perceived state of the artifacts. In situations where artifacts are not entirely transparent, decisions may be imperfect, leading to a potential decrease in value and an increase in risk.

The Scrum Artifacts are seven and each of these play a major role in improving the performance of a Scrum Team [11] [12].

1. Product Vision: articulates the overarching, long-term objective of your product. The purpose of this statement is to serve as a guiding principle and a constant reminder for all stakeholders involved in the product's creation. It helps to ensure a shared understanding of the ultimate goal the team is pursuing. Creating a vision enables the team to approach product design from a top-down perspective. The meaning of this is to begin with a high-level vision statement, which is then translated into a strategic guide and action plan known as the product roadmap. This strategic roadmap is subsequently translated into a tactical development strategy. Especially in project with a lot of teams involved that are working into the same objective for years, have a common vision helps everyone to have something to refer to as a reminder of what they are doing and obviously why. The last characteristics is that always the vision must come before working on the product plan.

2. Product Backlog: after the vision it is necessary to break down the final product into smaller features, which will make up a prioritized list, i.e the product backlog. These small features in an Agile terminology are often called as User Stories, encompass key attributes such as description, order, estimate, and value. The Product Owner is responsible for managing the content, accessibility, and prioritization of the Product Backlog. His role it’s very essential since the Product Backlog is an ongoing process, indeed it is called a living artifact. A lot of factors influence the Product Backlog, as changes in company requirements, market conditions and technology, for these reasons the Product Owner has to be ready for changing it on the fly. Adding clarity and details on the work, make it easier the estimations and this process is called refinement of the Product Backlog. Once the items in the Product Backlog are reviewed and refined, there will be declared ready for selection in a Sprint Planning meeting.

3. Sprint Vision is a brief description of what the team intends to accomplish during an agile sprint. The main feature is that has to be

a measurable objective, so that the team can, at any point in the sprint, easily understand whether it is deviating from the sprint goal or not. Sometimes happens that during the sprint some issues or new tasks arise throughout the sprint, the sprint vison provides a reference for the team on what to focus on and how to arrange or reprioritize various tasks. The Sprint Vision is usually built during the design of the next sprint by the team. It informs the scrum team on why they are investing their time, money and effort into the sprint.

4. Sprint Backlog consists of the items chosen from the Product Backlog, during the Sprint Planning. These items are called user story that consists of an informal explanation of a software system's features. Indeed, it's the team forecast of what functionality will be available in the next increment. The Sprint Backlog serves as a detailed plan for the team to monitor during the Daily Scrum. Throughout the Sprint, adjustments are made, and the backlog evolves as the team gains insights into the work needed for the Sprint Goal. This evolution occurs as new tasks arise, contributing to an ongoing refinement of the Sprint Backlog. The team has exclusive control over updating the Sprint Backlog during the Sprint, ensuring it remains a transparent, real-time overview of their intended work. However, modifications to the Sprint Backlog aren't unilaterally decided by the development team. Any changes require negotiation with the product owner, product manager, or Scrum Master before implementation. This communication is valuable as it enables a collaborative approach, allowing for adjustments without completely discarding processes. Instead, it encourages finding ways to break down tasks into more manageable components, facilitating easier completion.

5. Definition of Done (DoD): the DoD defines when all the elements of a user story in a sprint backlog have been completed. The team must agree of what the meaning of "done". Usually, there is a first draft DoD that is decided by the team during the first Sprint Planning. Then, the DoD could be adapted during the project, following an Agile way. The Definition of Done consists in a checklist of a series of actions that must be performed before marking the user stories as "done".

6. The Product increment is the sum of all the product backlog items accomplished during a sprint. Indeed, the increment consists into the release of all the features that are matching the DoD, once is the sprint is arrived at the end. At that time, the new increment should be a fully operational product, indicating its functionality is a fundamental requirement. Every increment adds to all previous increments and undergoes thorough testing to ensure seamless operation when integrated. Increment definitions should evolve to encompass additional reliability requirements as Scrum Teams mature. For any given product, a defined increment serves as a guiding standard for all related work.

7. Burndown Chart is a graphical representation of how quickly the team completes the user stories or items on the sprint backlog. As a result, a burndown chart shows the total effort versus the amount of work for a sprint. Each of these elements required estimation, which has story points as its units. What a story point represents varies from project to project, but usually a story point represents a man-day of work, so as to facilitate the team's estimation process, planning at the beginning of the sprint, and overseeing the progress of the sprint. As shown in the image, on the y-axis are the story points planned at the beginning of the sprint, while the x-axis represents the time course. The line on the graph starts from the y-axis, on the sum of the story points that are present in the sprint backlog. When an item conforms to DoD, the line goes down by the corresponding number of story points that are "Done." This provides a clear view of the number of story points remaining to complete what was set at the beginning of the sprint. An ideal burndown chart trend should be about a 45-degree line, but in reality, things are quite different. As can be seen from the chart for some days no story points were "burned down," and especially at some times the number of story points increased. This happens because during the sprint, the Sprint Backlog may undergo some changes, for example due to internal or external dependencies from the team.

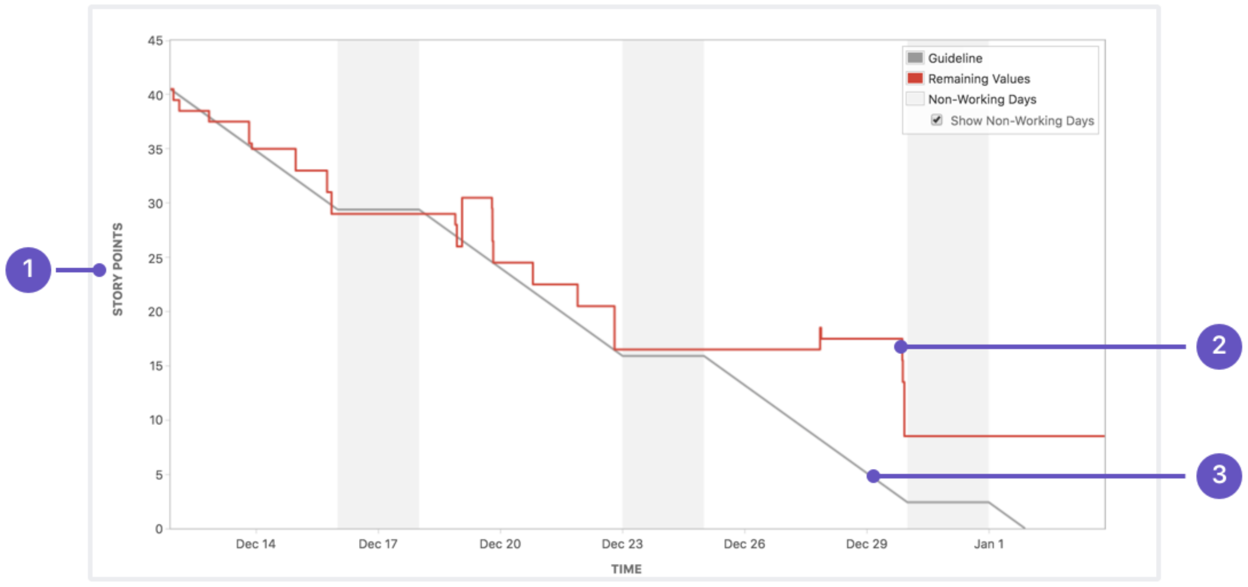


Figure 7: Burndown Chart

2.3.4 Scrum Cerimonies

Scrum meetings involve collaboration among the Scrum Master, Product Owner, and Development Team to plan tasks, review ongoing work, seek feedback, and perform other essential functions. It's important to note that not every Agile Scrum team has to do to all Scrum meetings, and a team need not strictly follow Scrum methodologies to benefit from these meetings each project has its own characteristics and specifications upon which different scrum meetings can be shaped. The following are the Agile Scrum meetings that can apply teams in various areas [13].

1. Sprint Planning: The items listed in a product backlog often span over several weeks or months, exceeding the scope of a single sprint. During Sprint Planning, collaboration among the Product Owner, development team, and Scrum Master is crucial to identify the most critical product backlog items for the upcoming Sprint. In this planning phase, the Product Owner and the development team align on a sprint goal, outlining what they aim to achieve in the impending sprint. The development team then assesses the sprint backlog, selecting the most valuable items feasible for completion within the sprint timeframe. Maintaining a sustainable pace, or working speed, is essential and should be chosen to ensure long-term sustainability. Estimating the effort required for each item, typically in working

days, is part of this process. For a sprint which is lasts one to four weeks, a Sprint Planning meeting should ideally be completed within two to eight hours. Various approaches exist for sprint planning, with one popular method involving a cyclical process: the team selects a product backlog item, ideally the next one in the Product Owner's prioritization and assesses its fit within the sprint, considering other planned items. This cycle repeats until the team's capacity for additional tasks is reached.

2. Daily Scrum / Daily Standup: each day throughout the sprint, ideally at a consistent time and location, team members gather for the Daily Scrum meeting, with a maximum duration of 15 minutes. Referred to as the Daily Stand-Up due to the practice of participants standing. The idea of standing, with the continual increase of projects in remote mode, of course, fails, but the basic idea is to keep the meeting brief. In the Daily Scrum, the main points to emerge at this meeting for each developer are: what progress I made since the last Daily Scrum? Are there any impediments that will not make me reach my goal? In this way any team member can get an overview on the tasks of the other developers and getting an insight if the team will be able to reach the sprint goals. For these reasons the Daily Scrum is vital for ensuring the team's agility and adaptability during the sprint. Importantly, the Daily Scrum does not serve as a platform for problem resolution. Instead, teams convene in smaller groups after the meeting to discuss and address issues. Essentially, the Daily Scrum involves checks, synchronizations, and adaptive planning, empowering a self-organized team to enhance overall performance.

3. Backlog Refinement: in the context of a product backlog involves an ongoing process to refine the main work, that is usually called epic and so add details such as description, order, and size. This continual process ensures that backlog items are refined into workable increments of value, enabling the team to deliver tangible outcomes each sprint. Understanding the "why" behind refinement is crucial. Unlike traditional big-requirements-upfront approaches, ongoing refinement prevents requirements from becoming meaningless and irrelevant. It also complements Sprint Planning by focusing on future sprints and maintaining a steady flow of work. Additionally, refinement is not solely the responsibility of the Product Owner; it requires alignment among stakeholders, developers, and the product owner to ensure that backlog items contribute to the overall product

goal. To refine effectively, it's essential to maintain focus on the product goal and ask the right questions to ensure that backlog items align with user needs and create value. Refinement should occur early and frequently to generate maximum value and keep the team on track towards achieving its objectives. While commonly associated with the Backlog Refinement Meeting, refinement is an ongoing activity rather than a singular event, emphasizing the importance of continual improvement and adaptation in Agile development.

4. Sprint Review: the conclusion of each sprint, two additional inspect-and adapt activities take place, one of which is known as the Sprint Review. The primary objective of this Scrum event is to assess the status of the work in progress product. For doing it, the primary focus is on evaluating recently completed features in the context of the overall development effort. This collective gathering provides everyone with a clear understanding of the current status and an opportunity to catch on the fly, the development for the optimal solution for the company. A successful Sprint Review facilitates a bidirectional flow of information. Individuals outside the Scrum Team gain insights into the product's status and can influence upcoming steps. Simultaneously, members of the Scrum Team enhance their comprehension of business and marketing aspects by receiving continuous feedback on whether the product development aligns with the goal of captivating customers and users. Therefore, the Sprint Review serves as a pre-planned occasion to scrutinize and refine a product. External individuals can conduct feature reviews during a Sprint, contributing feedback that aids the Scrum Team in achieving its Sprint goal.

5. Sprint Retrospective: The second crucial event at the conclusion of a Sprint is the Sprint Retrospective, typically conducted after the Sprint Review and before the subsequent Sprint Planning session. While the Sprint Review focuses on reviewing and adapting the product, the Sprint Retrospective provides an opportunity to evaluate and refine the team's processes. During the Sprint Retrospective, the development team, Scrum Master, and Product Owner collaborate to assess what practices are effective and what aspects require improvement within the Scrum framework. The primary emphasis lies on fostering continuous enhancement of the process, in order to continue improve as a team and Scrum practices inside the Team. By

the conclusion of the Sprint Retrospective, the Scrum Team should bring on the table meaningful set of improvement actions to implement in the next Sprint. Following the Sprint Retrospective, the cycle recommences, initiating with a new Sprint Planning session where the team identifies the most valuable tasks to prioritize and focus on.

3 Company and process overview

3.1 Company history

Reply is a multinational consulting, technology and innovation company specializing in providing services and solutions in a variety of industries, including telecommunications, media, financial services, healthcare, automotive, retail and logistics. The company offers a wide range of services, including digital transformation, IT consulting, software development, cybersecurity, cloud services and data analytics [14]. Founded in 1996 in Italy, Reply has grown into a global organization with offices and operations in Europe, the Americas, Asia Pacific and the Middle East. The company's headquarters are located in Turin, Italy.

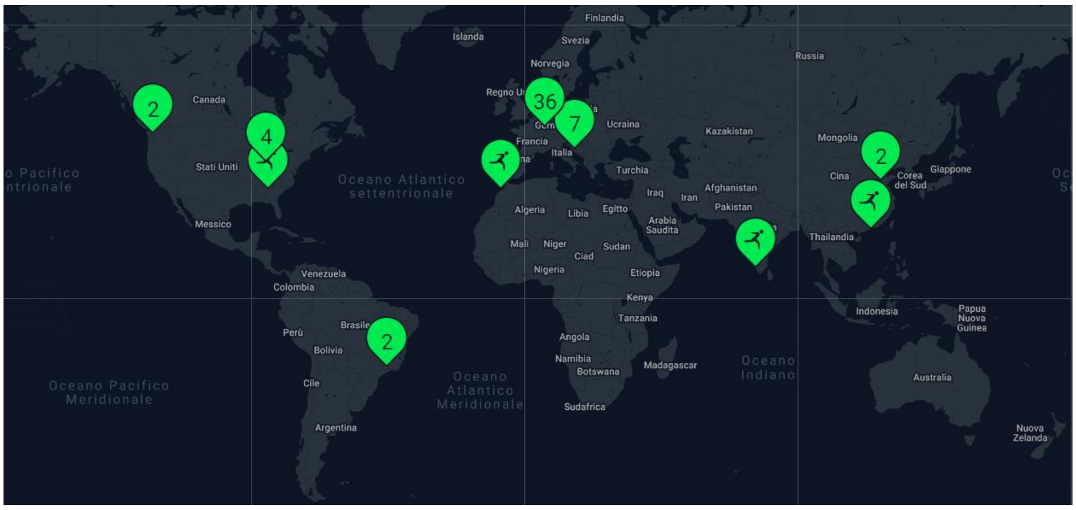


Figure 8: Reply's companies' location (https://www.reply.com/en)

Reply operates through a network of specialized companies, each of which focuses on specific industry or technology sectors. These companies work closely together to provide integrated solutions tailored to meet the diverse needs of customers around the world.

Over the years, Reply has established itself as a leader in digital transformation and innovation, leveraging cutting-edge technologies such as artificial intelligence, Internet of Things (IoT), blockchain, and machine learning to drive business success for its customers. The company has a proven track record of providing innovative solutions that help organizations adapt to rapidly changing market dynamics, improve operational efficiency, enhance customer experience, and stay ahead of the competition.

Reply's history is one of continuous growth and expansion, marked by strategic acquisitions, partnerships, and investments in emerging technologies. The company has received numerous awards and accolades for its innovative solutions, industry expertise, and commitment to customer service excellence.

Overall, Reply is recognized as a trusted partner for organizations seeking to navigate the complexities of the digital age and unlock new opportunities for growth and success. With a comprehensive portfolio of services, deep industry knowledge, and a global presence, Reply continues to be at the forefront of driving innovation and digital transformation in industries around the world.

3.2 Market Focus

Reply is a major player of big European industrial groups across various market segments, including Telco and Media, Banking, Insurance and Financial companies, Industry and Services, Energy and Utilities, and Public Administration.

In the Telco and Media sector, Reply holds a leading position as a technological partner. They specialize in redefining omni-channel engagement models, integrating physical and digital touchpoints, and developing innovative customer experience solutions, often integrated with social models.

For Banks, Insurance Companies, and Financial Operators, Reply plays an increasingly active role in digital transformation. They collaborate with industry leaders to define comprehensive multi-channel customer journey and engagement strategies, spanning from digital branding to the overhaul of technological architectures.

In Manufacturing and Retail, Reply supports companies in transforming and managing their information systems. Their services cover strategic design, redefinition of core business processes, and solutions for application integration. They offer a specific service offering for the retail sector, combining e-commerce and multi-channel consulting to create engaging customer experiences across various platforms.

Within Energy and Utilities, Reply leverages its deep understanding of the market to assist companies in adapting to the ongoing transformation in energy generation, distribution, and sales. They offer solutions for smart metering, smart grid management, real-time pricing, and demand response, aiding in the development of new operational models.

In Government and Defense, Reply capitalizes on its experience in advanced online services to create vertical applications and expertise. This enables them to implement specific solutions for managing relationships with the public and businesses, focusing on cost-cutting while maintaining service quality and improving opportunities for consumers in the health and public administration sectors.

3.3 The Project

This chapter will delve into the execution of the "Connected Car Services" project, completed during the collaboration between Reply and a company which is a German leader in the automotive sector.

First, an overview of what a connected car means is given, so as to get into the context of the product that the project focuses on.

3.3.1 Context: Connected car

A connected car, also known as a smart car or internet-enabled car, is a vehicle that is equipped with internet connectivity and often has built-in sensors and communication systems. These features allow the car to communicate with other devices, networks, and services, both inside and outside the vehicle.

One of the key features of connected cars is the ability to provide various services and functionalities to enhance safety, convenience, and efficiency.

In the following paragraph are some key components and capabilities of connected cars:

1. Internet Connectivity: Connected cars are equipped with internet connectivity, allowing them to exchange data with external servers, cloud platforms, and other vehicles. This connectivity can be established through cellular networks, Wi-Fi, or other communication technologies.

2. Telematics: Telematics systems in connected cars gather data related to the vehicle's performance, location, and other parameters. This data can be transmitted to manufacturers, service providers, or third-party applications for analysis and use in various applications, such

as predictive maintenance, insurance purposes, and fleet management.

3. In-Vehicle Infotainment (IVI): Connected cars often feature advanced infotainment systems that provide entertainment, navigation, communication, and other services to occupants. These systems typically include touchscreen displays, voice recognition, and integration with smartphones to access music, apps, and other content.

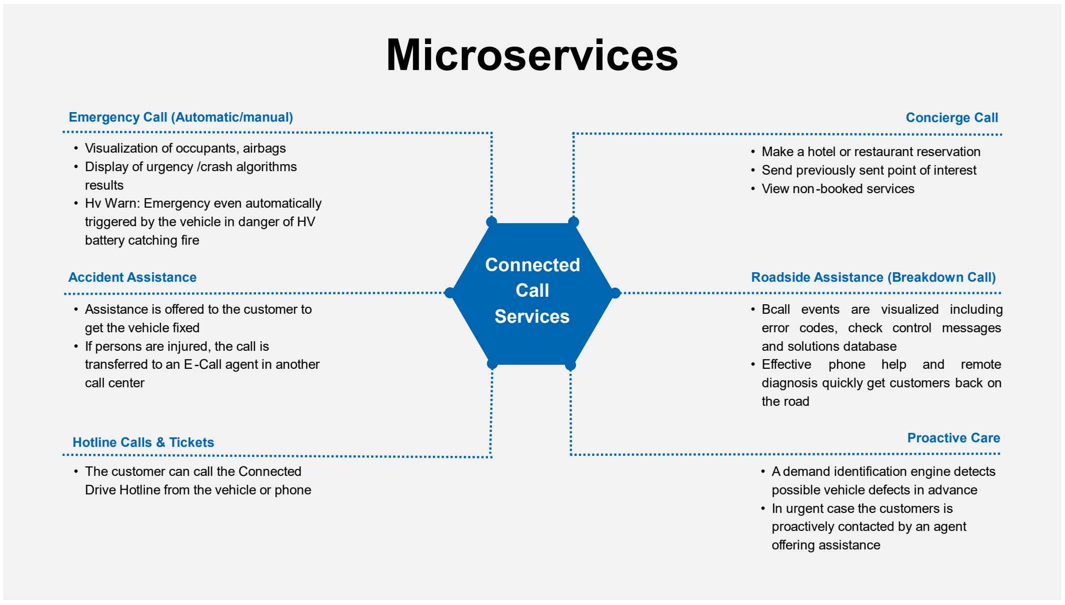
4. Safety and Security: Connected cars may incorporate features aimed at enhancing safety and security, such as automatic emergency calling (eCall), remote vehicle tracking, stolen vehicle recovery, and remote diagnostics. These features can help improve response times in emergencies and assist in the recovery of stolen vehicles.

5. Vehicle-to-Vehicle (V2V) Communication: Connected cars can communicate with each other using V2V communication technology, sharing information about their speed, position, and other relevant data. This capability enables advanced safety features, such as collision avoidance systems and cooperative adaptive cruise control, by allowing vehicles to anticipate and react to potential hazards on the road.

6. Vehicle-to-Infrastructure (V2I) Communication: In addition to communicating with other vehicles, connected cars can also exchange data with roadside infrastructure, such as traffic lights, road signs, and toll booths. This enables features like real-time traffic updates, traffic signal optimization, and electronic toll collection, which can improve traffic flow and overall driving efficiency.

3.3.2 Scope of the Project

Connected Call Services enables the customer to be automatically (vehicle-triggered) or manually (user-driven) connected to a call center agent in situations, where the customer needs assistance or information in the context of his vehicle and the ConnectedDrive services of his vehicle. It combines real time data, the voice connectivity from the vehicle and a call center application to provide a premium customer experience. Connected Call Services enables customer support by handling a series of calls for multiple stakeholders. There are 6 different call/event types that reflect different use cases and scenarios:



Emergency Call (Automatic/manual)
- Visualization of occupants, airbags
- Display of urgency /crash algorithms results
- Hv Warn: Emergency even automatically triggered by the vehicle in danger of HV battery catching fire

Accident Assistance
- Assistance is offered to the customer to get the vehicle fixed
- If persons are injured, the call is transferred to an E-Call agent in another call center

Hotline Calls & Tickets
- The customer can call the Connected Drive Hotline from the vehicle or phone

Concierge Call
- Make a hotel or restaurant reservation
- Send previously sent point of interest
- View non-booked services

Roadside Assistance (Breakdown Call)
- Bcall events are visualized including error codes, check control messages and solutions database
- Effective phone help and remote diagnosis quickly get customers back on the road

Proactive Care
- A demand identification engine detects possible vehicle defects in advance
- In urgent case the customers is proactively contacted by an agent offering assistance

Figure 9: Microservices covered by the project

- Emergency call: it can be triggered automatically, indeed, in many modern vehicles, the eCall system is automatically triggered when sensors detect a severe accident, such as a collision that deploys airbags. When such an event occurs, the system automatically dials the emergency services number and sends relevant data, such as the vehicle's location and potentially other diagnostic information, to the emergency response center.
  Additionally, drivers or passengers can manually activate the eCall system by pressing a dedicated button. This is useful in situations where immediate assistance is needed but no severe accident has occurred, such as attending a medical emergency or an accident involving another vehicle.
  Once the emergency call is connected, the vehicle occupants can communicate directly with the emergency services call center to provide information about the situation, the number of occupants in the vehicle, and

any injuries or medical conditions that require attention. One of the key features of eCall systems is the transmission of the vehicle's location to the emergency services center. This allows responders to quickly locate the vehicle, even if the occupants are unable to provide their location verbally. In addition, the customer will be connected directly to a call center agent who speaks his native language, even if he is located outside their country, in order to facilitate communication in an emergency situation.
Overall, eCall systems are designed to expedite emergency response times, potentially saving lives by ensuring that help arrives quickly in critical situations on the road. They are becoming increasingly common in new vehicles as a standard safety feature.

- Accident call: it is another type of call, that is manually triggered in a situation in which the customer needs assistance to get his vehicle fixed. Once is triggered, the customer will be connected to an agent in the call center who will be available to support the customer in any way possible. So, if after the accident the car can be driven, the status of the car, catched by sensors on the car, is directly transmitted to the preferred shop set by the customer. Instead, in the case where the car cannot be driven, the accident assistant agent will arrange a recovery of the car. Of course, in case a vehicle component is injured, the call is transferred to an emergency call center agent.

- Roadside assistance (Breakdown call): In the event of a breakdown or malfunction of the car, the customer can manually, triggered the breakdown call. The premium feature of this service is that at the press of the button, the call center agent, will have available on his display a current diagnostic of the car, in order to reduce the time of intervention and not have to try to understand the problem charged by the customer, from his explanation, which in times of emergency can be confusing and not explanatory. The call center agent in this case, is a specialist able to remotely diagnose the car as if it were in the workshop.

- Concierge call: at the touch of a button, the driver will be put in touch with a call center operator, who will be available for suggesting a restaurant, booking an hotel room or everything is needed.
  Once the place is chosen, on the navigation system in the car will appear the track to follow for reaching the desired place.

- The Proactive Care System is a revolutionary vehicle maintenance system. Instead of waiting for problems to arise, this innovative system takes a proactive approach, constantly monitoring various aspects of the car's performance. Using advanced sensors and predictive analytics, it analyzes data to anticipate maintenance needs and identify potential issues before they become major problems. Through customized recommendations and timely alerts, it keeps the customer informed about upcoming service requirements, allowing the client to take proactive action to keep the vehicle in top condition. Especially, in urgent cases, the customer is proactively contacted by an agent of the call center.

3.3.3 Teams Involved

The entire project is under the leadership of Reply, with a focus on effective coordination and collaboration among the various teams involved. As part of the product breakdown, there are three main teams contributing to the project's success. These teams are located across different regions, with one team based in Poland, another in Turin, Italy, and the third in Milan, Italy. Each team brings unique expertise and capabilities to the table, in order to achieve the project's objectives and deliver a high-quality final product.

The subdivision is as follows:

1. Call Handling Backends: this team, based in Turin, is under my coordination as the Scrum Master. The backend's primary function is to receive call requests and generate corresponding events. These events are enriched with essential information retrieved from the vehicle, including customer details, vehicle information, location data, market specifics, and more. This comprehensive dataset ensures that all necessary information for routing the call and providing assistance to the customer is readily available. Subsequently, this information is transmitted to the voice connectivity component to establish the appropriate voice connection and forwarded it to the call center, where the event will be processed and managed.

2. Voice Connectivity: the team located in Milan specializes in managing the Voice Connection, facilitating the communication stream between customers in vehicles or on phones and agents in call centers. This involves implementing a robust routing logic to ensure that customers are directed to the appropriate call center based on market specifications. Additionally, the team ensures that customers are paired with agents fluent in their language and equipped to assist with specific requests such as emergencies, breakdowns, hotline support, and concierge services.

3. Call Handling Frontends: based in Poland, this team is responsible for developing the user interface utilized by call center agents and other user groups involved in handling calls. The level of integration with the call center software determines whether agents manually select the corresponding data event from a list or if the event automatically appears when the agent accepts the customer call.

   The frontends provided by this team present all relevant information pertaining to the event, the vehicle and its status, the customer, and the customer's situation. This information is transmitted via events provided by the Call Handling Backends. Additionally, these frontends offer tools to assist the customer and resolve issues remotely.

4 Project’s Plan

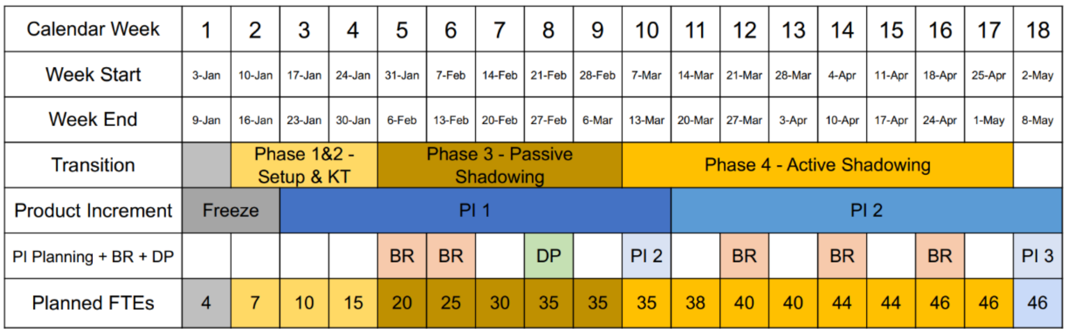
How to be agile

This project was the first one for me, using Agile methodology, my role as a Scrum Master involved conveying the values and processes of Scrum to the team. I quickly encountered the challenge of transitioning from theory to practice when implementing such an intricate framework.

From these difficulties rose up the idea of composing a thesis that could be use as a sort of guide for future projects at Reply. While each project possesses its unique characteristics and demands a tailored approach, the ensuing "Manual" draws upon my understanding of Agile and Scrum, my firsthand experience applying the framework to a complex international project, and, significantly, the literature that expanded my perspective, integrating metrics and processes previously unfamiliar to me.

4.1 Project’s first phase

The project on Reply's side officially commenced in January 2023. Given the critical nature of the product, it demands continuous maintenance and the concurrent development of new features to enhance vehicle connectivity and innovation, thereby delivering a premium experience to customers. As a critical and highly delicate handover, there was a transition phase from the outgoing company that lasted for 4 months, divided into 3 main phases: setup & KT, passive shadowing, and active shadowing.



Calendar Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Week Start	3-Jan	10-Jan	17-Jan	24-Jan	31-Jan	7-Feb	14-Feb	21-Feb	28-Feb	7-Mar	14-Mar	21-Mar	28-Mar	4-Apr	11-Apr	18-Apr	25-Apr	2-May
Week End	9-Jan	16-Jan	23-Jan	30-Jan	6-Feb	13-Feb	20-Feb	27-Feb	6-Mar	13-Mar	20-Mar	27-Mar	3-Apr	10-Apr	17-Apr	24-Apr	1-May	8-May
Transition		Phase 1&2 - Setup & KT			Phase 3 - Passive Shadowing					Phase 4 - Active Shadowing								
Product Increment	Freeze		PI 1								PI 2							
PI Planning + BR + DP					BR	BR		DP		PI 2		BR		BR		BR		PI 3
Planned FTEs	4	7	10	15	20	25	30	35	35	35	38	40	40	44	44	46	46	46

Figure 10: Initial project phases

During the setup & KT phase, which lasted for 3 weeks, efforts were made to ensure that all components of Reply had access to the systems and tools managed by the client (such as Jira, Confluence, etc.) necessary for project work. Regarding the KT part, comprehensive video lectures were conducted by both the client and some members of the outgoing company. The client presented general aspects common to all three teams, while representatives from the outgoing company presented application details to the dedicated team to understand its actual functioning.

The passive shadowing phase, lasting 5 weeks, saw the new teams passively participating in all project processes during the initial 2-3 weeks. This allowed for a clear understanding of the client's expectations once Reply became the sole company on the project. While this phase certainly had benefits in terms of project flow, it also made it challenging to change certain stagnant processes to which the client was accustomed. In the latter weeks of this phase, Reply teams worked actively alongside the outgoing company, but the product responsibility still lay with the latter.

During the active shadowing phase, only a few members of the outgoing teams remained on the project, but all product responsibility had legally transitioned to Reply. Although the teams in this phase did not fully engage in new implementations, they began taking ownership of every project phase. By the end of this final project phase in late April, Reply became the sole company on the project.

4.2 High level process overview

In contrast to other projects where the product is developed from scratch, this project involves an ongoing development of new implementations aimed at enhancing the features of the final product, thereby offering new functionalities to the client or improving existing systems and architecture. Additionally, there is a significant component of activities dedicated to maintenance and application updates, essential for ensuring the smooth functioning of the software.

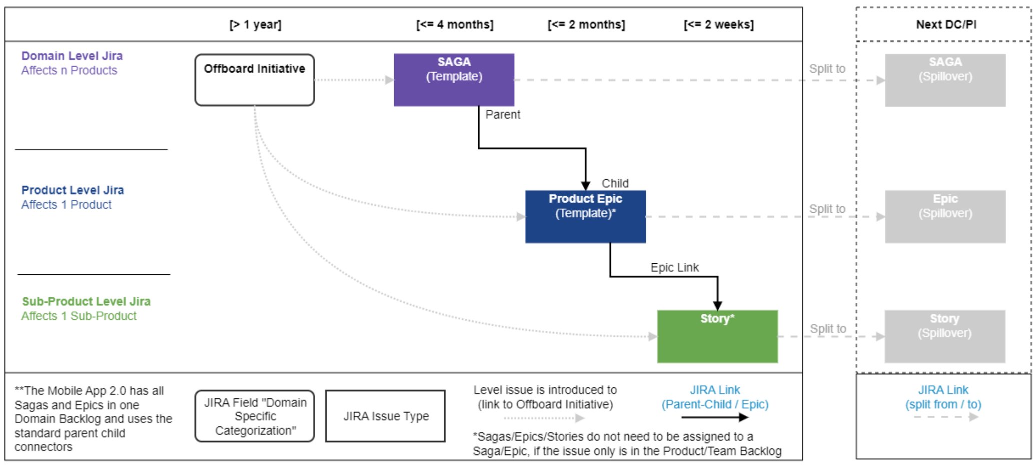

[> 1 year]
[<= 4 months]
[<= 2 months]
[<= 2 weeks]
Next DC/PI
Domain Level Jira
Affects n Products
Product Level Jira
Affects 1 Product
Sub-Product Level Jira
Affects 1 Sub-Product
Offboard Initiative
SAGA
(Template)
Parent
Child
Product Epic
(Template)*
Epic Link
Story*
Split to
Split to
Split to
SAGA
(Spillover)
Epic
(Spillover)
Story
(Spillover)
**The Mobile App 2.0 has all Sagas and Epics in one Domain Backlog and uses the standard parent child connectors
JIRA Field "Domain Specific Categorization"
JIRA Issue Type
Level issue is introduced to (link to Offboard Initiative)
JIRA Link (Parent-Child / Epic)
*Sagas/Epics/Stories do not need to be assigned to a Saga/Epic, if the issue only is in the Product/Team Backlog
JIRA Link (split from / to)

Figure 11: : Saga-Epic-Story

Saga

The high-level project is directly managed by the client company. At the level of business requirements, which involve new features impacting the end customer seated in the new vehicle, decisions are made directly by the automotive manufacturer. Business requirements at the project level are translated into Sagas, which are high-level requirements impacting multiple products. Indeed, the main characteristics of a Saga are as follows: it represents a medium to long-term strategy (up to 4 months), and it is composed of at least 2 epics. Project teams are not included in this decision-making process.

Epic

Moving down to a lower level, we have epics, which are the ownership of different Product Owners. The maximum timeframe available to complete an epic is two months, which corresponds to the length of a Product Increment (PI).

The main characteristics of an epic are as follows:

- An epic is a large body of work that can be broken down into smaller tasks or user stories.
- Epics represent high-level features or initiatives that are too big to be completed in a single iteration or sprint.
- They often span multiple sprints and are broken down into smaller, more manageable units of work called user stories.

- Epics provide a way to organize and prioritize work, allowing teams to focus on delivering value incrementally while keeping sight of the larger project goal.

- Possible example of epics into the project are implementation of a new type of call from the vehicle or migration to a different type of DB for one application.

User Stories

Finally, there are User Stories, the smallest package into which an epic is subdivided and upon which the development team will work. To conclude an epic, it is necessary to complete all the user stories that are associated with that epic.

The main characteristics of a user stories are as follows:

- In Agile development, a user story is a small, self-contained unit of work that represents a single piece of functionality from the end user's perspective.

- Stories describe the desired outcome or behavior of a feature or functionality, often written in a format like "As a [user], I want [feature] so that [benefit].

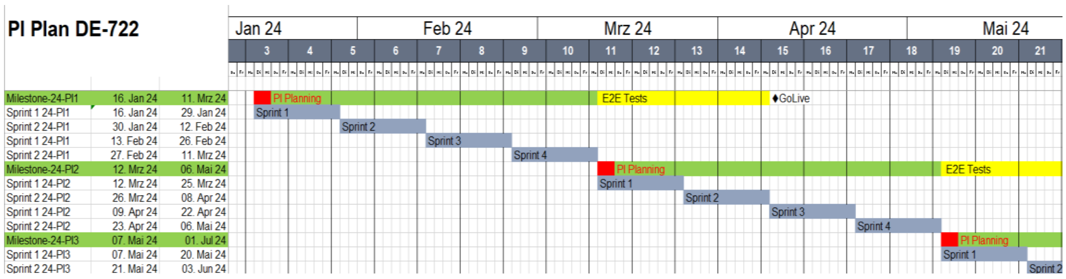
- They are typically written collaboratively by the development team and stakeholders and are used to capture requirements, facilitate discussions, and prioritize work.

- User stories are often organized and tracked on a team's backlog and are pulled into sprints or iterations for implementation.

Into this type of project as a backend team is more difficult to describe a user story by the point of view of the final user, so the title are often more with technical requirement than business requirement.

4.3 Timeline of the project

The project, conducted using the Scaled Agile Framework (SAFe), is divided into Product Increment (PI) each spanning a duration of 2 months, and it is divided into 4 sprints each having a duration of 2 weeks. The first two days of each PI are dedicated to the most crucial meeting of the entire increment, known as the PI Planning. During this meeting, each team initially presents what they have implemented and delivered in the previous PI, along with lessons learned and major challenges encountered. Additionally, teams specify if any epics planned in the previous PI were not delivered, thereby informing other teams that may have dependencies on these epics.

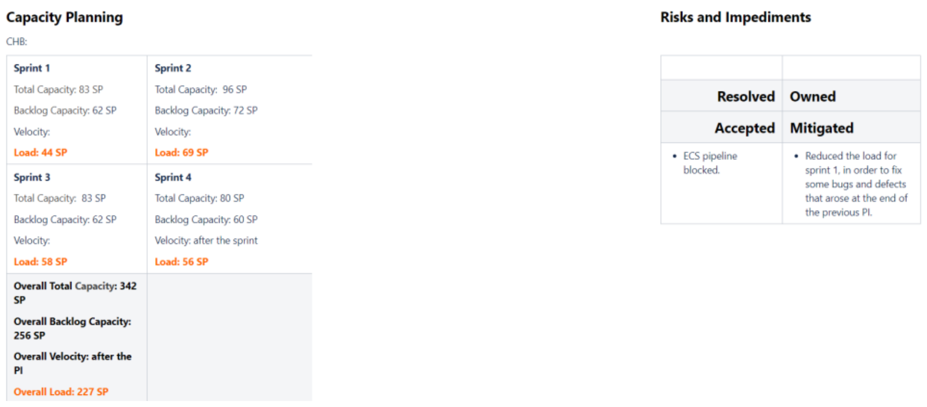


PI Plan DE-722

Milestone-24-PI1	16. Jan 24	11. Mrz 24
Sprint 1 24-PI1	16. Jan 24	29. Jan 24
Sprint 2 24-PI1	30. Jan 24	12. Feb 24
Sprint 1 24-PI1	13. Feb 24	26. Feb 24
Sprint 2 24-PI1	27. Feb 24	11. Mrz 24
Milestone-24-PI2	12. Mrz 24	06. Mai 24
Sprint 1 24-PI2	12. Mrz 24	25. Mrz 24
Sprint 2 24-PI2	26. Mrz 24	08. Apr 24
Sprint 1 24-PI2	09. Apr 24	22. Apr 24
Sprint 2 24-PI2	23. Apr 24	06. Mai 24
Milestone-24-PI3	07. Mai 24	01. Jul 24
Sprint 1 24-PI3	07. Mai 24	20. Mai 24
Sprint 2 24-PI3	21. Mai 24	03. Jun 24

Figure 13: Timeline of the Project

After this initial activity, the teams split up and work separately to plan the next two months of work. During this phase, each epic in the backlog is discussed, and possible dependencies on other epics or external teams are analyzed. Once this analysis is completed, each story within the epic is selected and placed into the sprint until the maximum capacity is reached (this aspect will be detailed in the Planning phase). On the second day, another plenary meeting takes place, during which each team presents the output of the first day: the planning board and program board.



Capacity Planning

CHB:

Sprint 1 Total Capacity: 83 SP Backlog Capacity: 62 SP Velocity: **Load: 44 SP**	**Sprint 2** Total Capacity: 96 SP Backlog Capacity: 72 SP Velocity: **Load: 69 SP**
Sprint 3 Total Capacity: 83 SP Backlog Capacity: 62 SP Velocity: **Load: 58 SP**	**Sprint 4** Total Capacity: 80 SP Backlog Capacity: 60 SP Velocity: after the sprint **Load: 56 SP**
Overall Total Capacity: 342 SP **Overall Backlog Capacity: 256 SP** **Overall Velocity: after the PI** **Overall Load: 227 SP**	

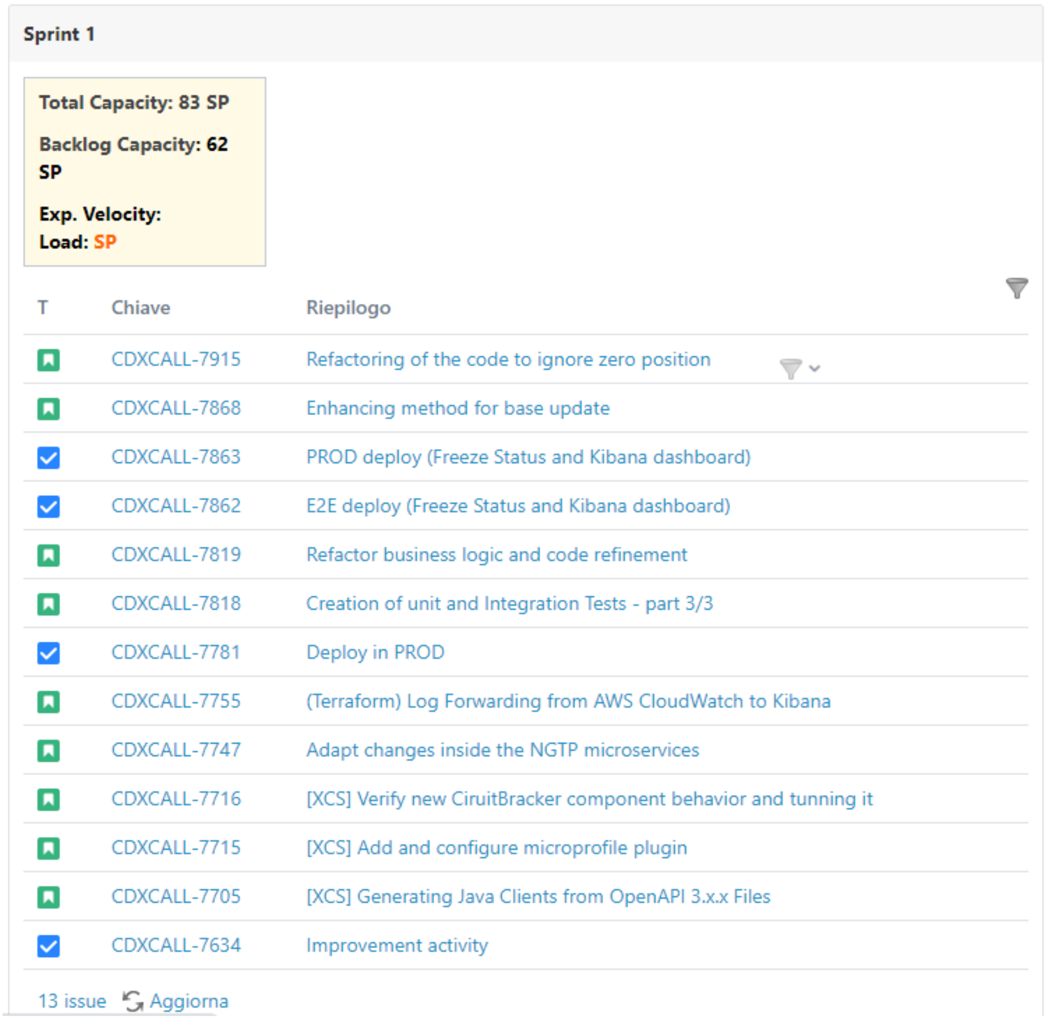
Risks and Impediments

Resolved	**Owned**
Accepted	**Mitigated**
• ECS pipeline blocked.	• Reduced the load for sprint 1, in order to fix some bugs and defects that arose at the end of the previous PI.

Figure 12: The Planning Board

The planning board is a table where epics are categorized into seven groups based on their ultimate purpose. Additionally, it provides an overview of the team's capacity for each sprint and a risk matrix where the team identifies potential risks related to the work, they will undertake in the following two months.

The program Board goes more into details, providing an overview on the sprint in which the epics will be delivered and how each sprint is fulfilled at user story level, as it is possible to see in the image below.



Sprint 1

Total Capacity: 83 SP

Backlog Capacity: 62 SP

Exp. Velocity:
Load: SP

T	Chiave	Riepilogo
🟩	CDXCALL-7915	Refactoring of the code to ignore zero position
🟩	CDXCALL-7868	Enhancing method for base update
✓	CDXCALL-7863	PROD deploy (Freeze Status and Kibana dashboard)
✓	CDXCALL-7862	E2E deploy (Freeze Status and Kibana dashboard)
🟩	CDXCALL-7819	Refactor business logic and code refinement
🟩	CDXCALL-7818	Creation of unit and Integration Tests - part 3/3
✓	CDXCALL-7781	Deploy in PROD
🟩	CDXCALL-7755	(Terraform) Log Forwarding from AWS CloudWatch to Kibana
🟩	CDXCALL-7747	Adapt changes inside the NGTP microservices
🟩	CDXCALL-7716	[XCS] Verify new CiruitBracker component behavior and tunning it
🟩	CDXCALL-7715	[XCS] Add and configure microprofile plugin
🟩	CDXCALL-7705	[XCS] Generating Java Clients from OpenAPI 3.x.x Files
✓	CDXCALL-7634	Improvement activity

13 issue Aggiorna

Figure 14: Objectives and Planning Board

4.4 Deployment

After the PI Planning meeting, the deploy in E2E can take place. During this phase what was implemented in the previous two months can be released in E2E environment. End-to-end environment refers to a setup or configuration that mirrors the production environment as closely as possible. During this phase the tests are triggered directly from the vehicle in order to replicate the entire flow of a software application or system, from the user interface to the back-end systems and databases. Having an end-to-end environment is crucial for testing the full functionality and integration of a software system in a realistic setting before it is deployed to production. It allows developers and testers to identify and address any issues or inconsistencies that may arise when different components interact with each other in a production-like environment. This type of testing helps ensure that the software behaves as expected and meets the requirements and expectations of end-users.

This testing phase lasts for one month, and after that, if there are no critical bugs present that could cause issues for the end customer, the product can be deemed stable, and new features can be released into the production environment, via another deployment.

5 Agile processes into the project

5.1 Backlog Refinement

Backlog refinement, also known as backlog grooming, is an essential activity in Agile methodologies such as Scrum. During backlog refinement, the product development team collaborates to review, prioritize, and refine product backlog items.

In the project we have two scheduled meetings where the refinement activity takes place. This meeting is attended by the whole development team and the POs who are two figures from the client company.

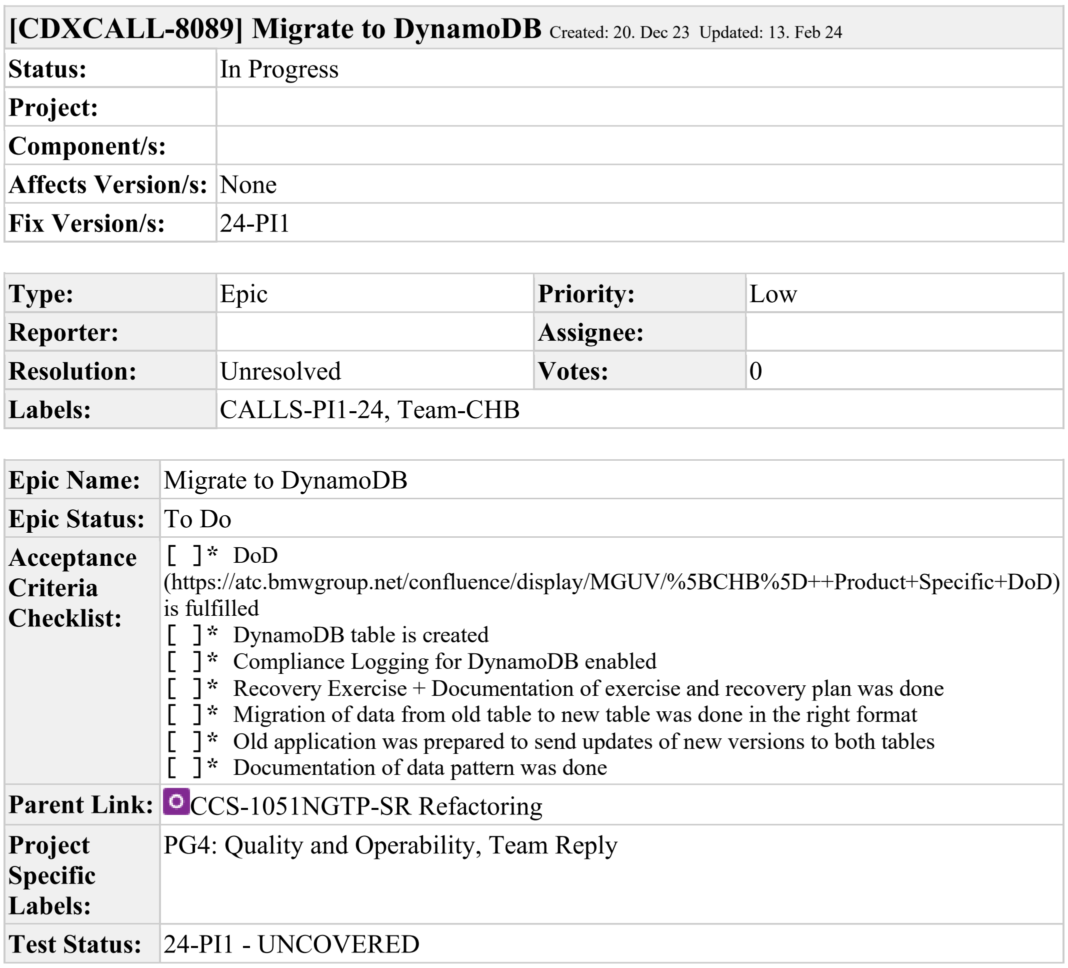
The main activities that happen at these meetings are as follows:

- Presentation of new epics, which refers to a large body of work. As is it possible to see in the image 15, an epic consists of a list of acceptance criteria that summarize the main points that need to be met to consider the epic done. While in the requirements part, the subtasks to be completed to achieve the final goal are described.

- During the presentation, several discussions occur about possible ways to implement the epic, possible dependencies on other teams or partner systems outside the project. The goal is to be clear about the goal to be achieved, so that later, the development team can study a strategy to implement the epic by dividing it into smaller bricks called stories.

- The refinement meeting is also the ideal time to present the epics that have been analysed or 'groomed', in which case the customer is presented with the chosen strategy for implementation and an estimate in Story Points and the corresponding T-Shirt Size that must be accepted before starting to implement the chosen solution.

- Finally, backlog refinement provides an opportunity for everyone to contribute their perspectives, ask questions, and align on the work ahead. Especially, the team can present possible improvements that should be done in order to improve some functionalities and in addition provides an opportunity for the team to ask questions and seek clarification on backlog items. This ensures that everyone has a clear understanding of what needs to be done.



[CDXCALL-8089] Migrate to DynamoDB Created: 20. Dec 23 Updated: 13. Feb 24

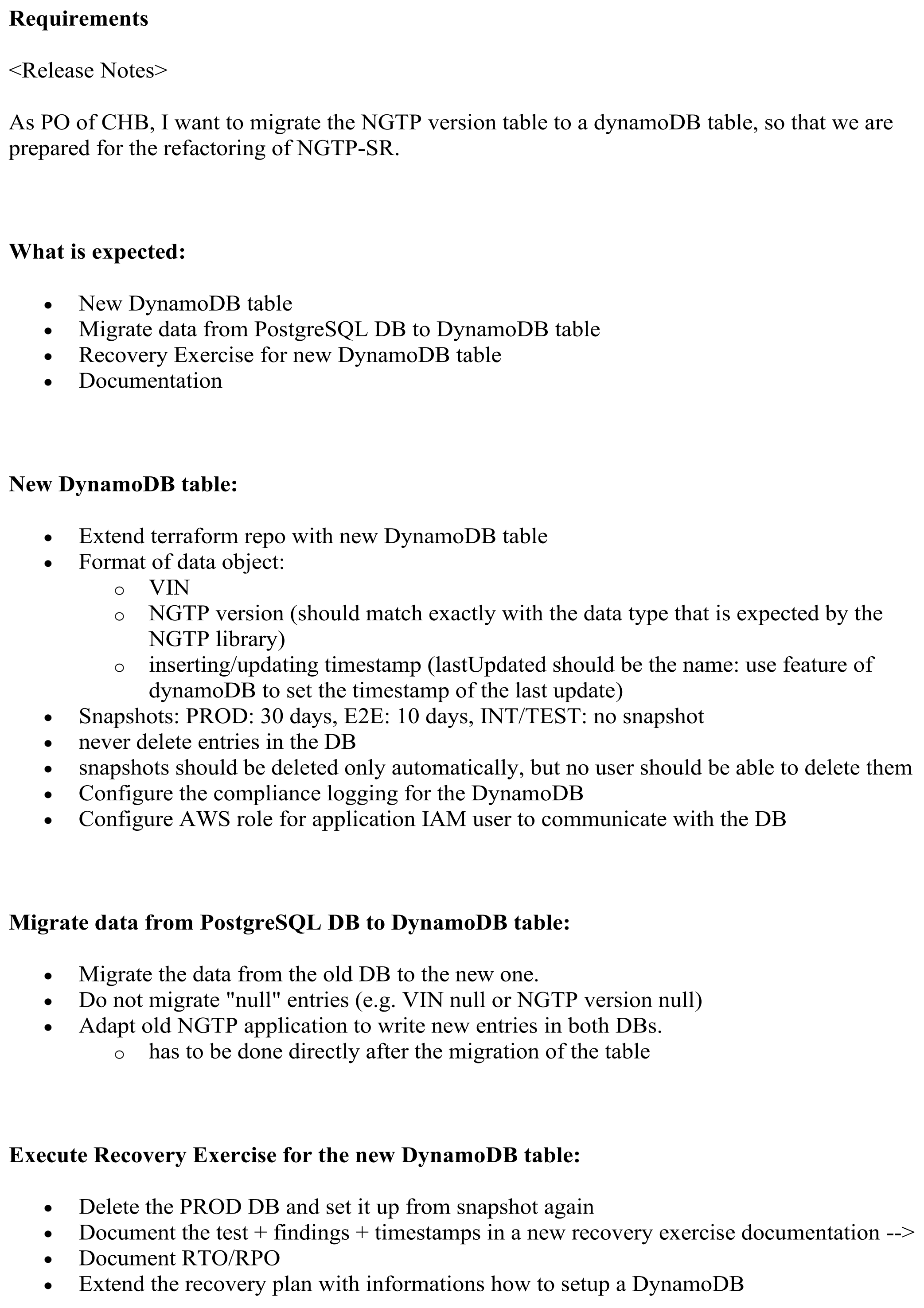
Status:	In Progress
Project:	
Component/s:	
Affects Version/s:	None
Fix Version/s:	24-PI1

Type:	Epic	**Priority:**	Low
Reporter:		**Assignee:**	
Resolution:	Unresolved	**Votes:**	0
Labels:	CALLS-PI1-24, Team-CHB		

Epic Name:	Migrate to DynamoDB
Epic Status:	To Do
Acceptance Criteria Checklist:	[ ]* DoD (https://atc.bmwgroup.net/confluence/display/MGUV/%5BCHB%5D++Product+Specific+DoD) is fulfilled [ ]* DynamoDB table is created [ ]* Compliance Logging for DynamoDB enabled [ ]* Recovery Exercise + Documentation of exercise and recovery plan was done [ ]* Migration of data from old table to new table was done in the right format [ ]* Old application was prepared to send updates of new versions to both tables [ ]* Documentation of data pattern was done
Parent Link:	CCS-1051NGTP-SR Refactoring
Project Specific Labels:	PG4: Quality and Operability, Team Reply
Test Status:	24-PI1 - UNCOVERED

Figure 15: Epic Document

Overall, backlog refinement is a collaborative and iterative process that helps ensure the product backlog remains in good shape and ready for development. It contributes to the success of Agile projects by providing clarity, focus, and flexibility in delivering value to customers.



Requirements

<Release Notes>

As PO of CHB, I want to migrate the NGTP version table to a dynamoDB table, so that we are prepared for the refactoring of NGTP-SR.

What is expected:

- New DynamoDB table
- Migrate data from PostgreSQL DB to DynamoDB table
- Recovery Exercise for new DynamoDB table
- Documentation

New DynamoDB table:

- Extend terraform repo with new DynamoDB table
- Format of data object:
    - VIN
    - NGTP version (should match exactly with the data type that is expected by the NGTP library)
    - inserting/updating timestamp (lastUpdated should be the name: use feature of dynamoDB to set the timestamp of the last update)
- Snapshots: PROD: 30 days, E2E: 10 days, INT/TEST: no snapshot
- never delete entries in the DB
- snapshots should be deleted only automatically, but no user should be able to delete them
- Configure the compliance logging for the DynamoDB
- Configure AWS role for application IAM user to communicate with the DB

Migrate data from PostgreSQL DB to DynamoDB table:

- Migrate the data from the old DB to the new one.
- Do not migrate "null" entries (e.g. VIN null or NGTP version null)
- Adapt old NGTP application to write new entries in both DBs.
    - has to be done directly after the migration of the table

Execute Recovery Exercise for the new DynamoDB table:

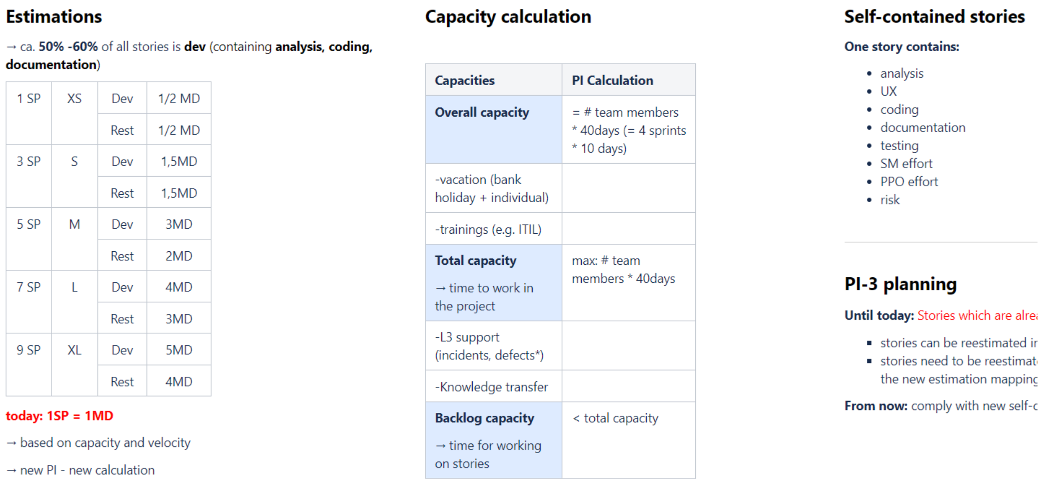
- Delete the PROD DB and set it up from snapshot again
- Document the test + findings + timestamps in a new recovery exercise documentation -->
- Document RTO/RPO
- Extend the recovery plan with informations how to setup a DynamoDB

Figure 16: Grooming Document

5.1.1 Estimation

Estimation in Agile software development refers to the process of predicting the time, effort, and resources required to complete a given user story, or backlog item. Estimation plays a crucial role in planning and prioritizing work, managing expectations, and facilitating communication within Agile teams. As a crucial topic, there were several discussions at the beginning of the project, and it took several months before reaching an agreement.

At the end the deal put in place is that the team estimates the user stories in Story Points and the estimation reflects the time of analysis and coding that a developer will spend for implementing such user story. Specifically, 1 Story point represents 1 day of work for one developer (1 story point = 1 man day). Then, through the conversion table, the team arrives at the final estimate, which is then presented to the client. For example, a user story estimated at 3 days will be sold to the client as a size M story, equivalent to 5 Story Points. This conversion is made because each user story carries associated risks: firstly, there is no certainty that the implementation will require exactly the time contained in the estimation. Additionally, it is done to compensate for the work of managing a Scrum team, especially the Scrum Master, who, due to his role, does not actively produce story points. And finally, to cover all the time spent during the deploy phase.



Estimations

→ ca. **50% -60%** of all stories is **dev** (containing **analysis, coding, documentation**)

1 SP	XS	Dev	1/2 MD
		Rest	1/2 MD
3 SP	S	Dev	1,5MD
		Rest	1,5MD
5 SP	M	Dev	3MD
		Rest	2MD
7 SP	L	Dev	4MD
		Rest	3MD
9 SP	XL	Dev	5MD
		Rest	4MD

today: 1SP = 1MD

→ based on capacity and velocity

→ new PI - new calculation

Capacity calculation

Capacities	PI Calculation
Overall capacity	= # team members * 40days (= 4 sprints * 10 days)
-vacation (bank holiday + individual)	
-trainings (e.g. ITIL)	
Total capacity → time to work in the project	max: # team members * 40days
-L3 support (incidents, defects*)	
-Knowledge transfer	
Backlog capacity → time for working on stories	< total capacity

Self-contained stories

One story contains:

- analysis
- UX
- coding
- documentation
- testing
- SM effort
- PPO effort
- risk

PI-3 planning

Until today: Stories which are alre

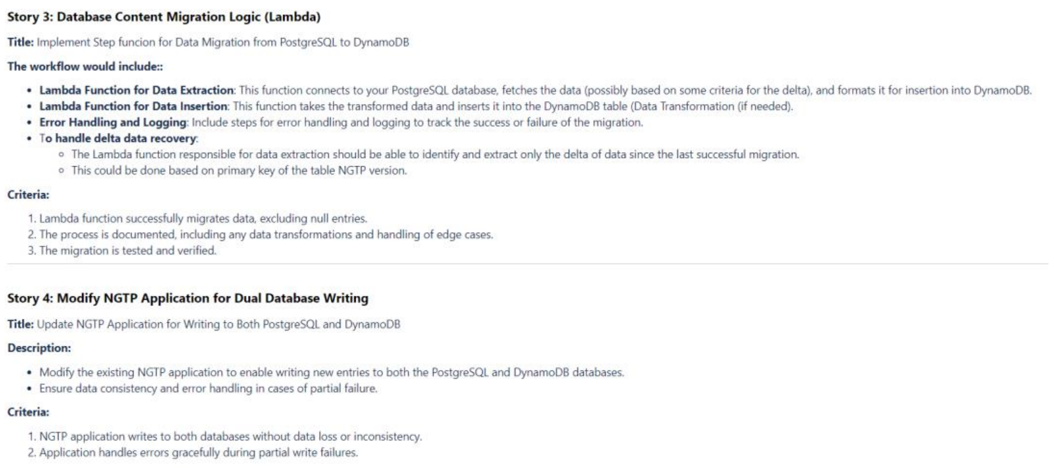
- stories can be reestimated ir
- stories need to be reestimat the new estimation mappinc

From now: comply with new self-c

Figure 17: Estimation agreement

5.1.2 Planning Poker

Initially, the estimates were made by the team's Tech Leader and the Scrum Master. Each team member had epics to analyze and study the best implementation solution for the epic. Once the analysis was complete, a document (e.g., an image 18) was produced containing the user stories necessary to complete the epic and a description of each, making the chosen resolution path visible to the client. Additionally, each user story was associated with a rough estimate. Once this document was produced, the Scrum Master and the Tech Leader would meet separately to discuss the analysis and confirm or modify the estimates, thus making the analysis definitively presentable to the client before it could be implemented.



Story 3: Database Content Migration Logic (Lambda)

Title: Implement Step funcion for Data Migration from PostgreSQL to DynamoDB

The workflow would include::

- **Lambda Function for Data Extraction**: This function connects to your PostgreSQL database, fetches the data (possibly based on some criteria for the delta), and formats it for insertion into DynamoDB.
- **Lambda Function for Data Insertion**: This function takes the transformed data and inserts it into the DynamoDB table (Data Transformation (if needed).
- **Error Handling and Logging**: Include steps for error handling and logging to track the success or failure of the migration.
- T**o handle delta data recovery**:
  - The Lambda function responsible for data extraction should be able to identify and extract only the delta of data since the last successful migration.
  - This could be done based on primary key of the table NGTP version.

Criteria:

1. Lambda function successfully migrates data, excluding null entries.
2. The process is documented, including any data transformations and handling of edge cases.
3. The migration is tested and verified.

Story 4: Modify NGTP Application for Dual Database Writing

Title: Update NGTP Application for Writing to Both PostgreSQL and DynamoDB

Description:

- Modify the existing NGTP application to enable writing new entries to both the PostgreSQL and DynamoDB databases.
- Ensure data consistency and error handling in cases of partial failure.

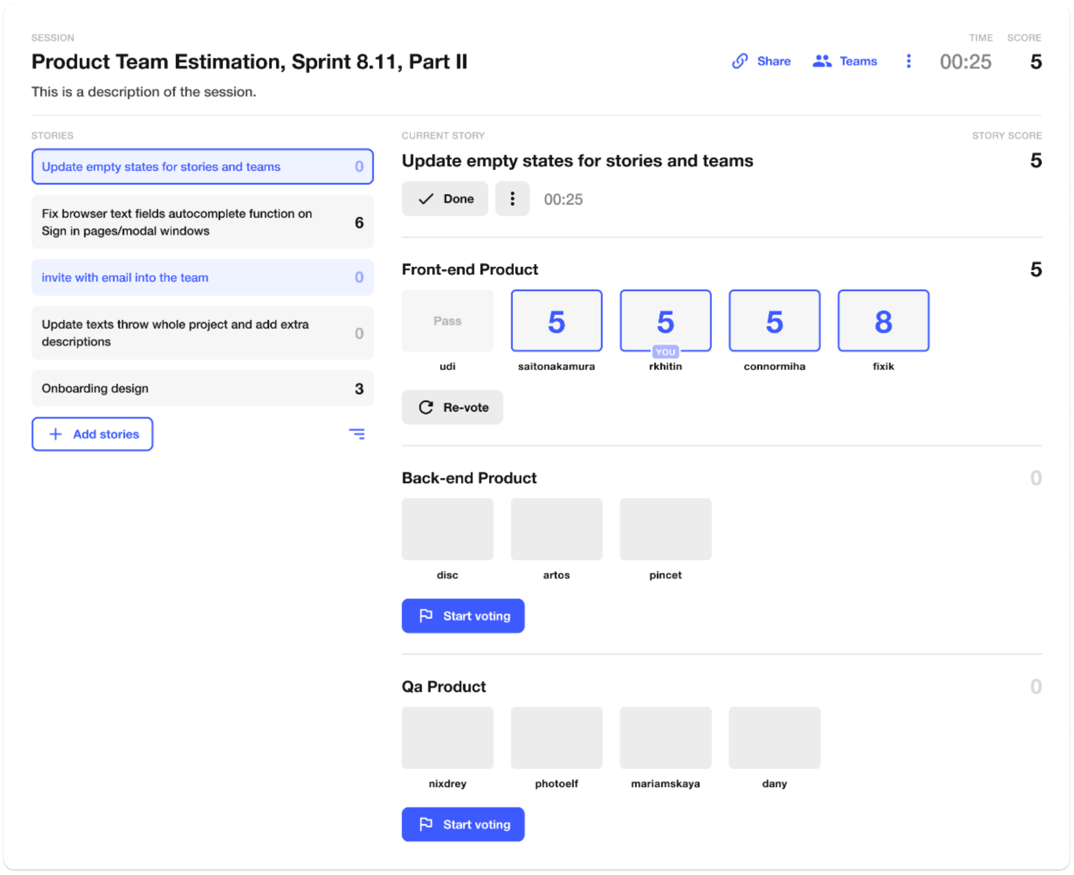
Criteria:

1. NGTP application writes to both databases without data loss or inconsistency.
2. Application handles errors gracefully during partial write failures.

Figure 18: User Stories description

The process had become well-established, and except for rare cases, there was not a significant difference observed between the actual value and the estimated value. However, at certain points in the project, the workload on the shoulders of the Tech Leader and the Scrum Master became too high, and knowledge was not widely spread within the team. For these reasons, I sought a new methodology in the literature to estimate the user stories [15].

Currently, the estimation phase inside the team is conducted by the Scrum Master, through the Planning Poker strategy, which is a collaborative estimation technique used in Agile software development to determine the relative size or effort required to complete user stories. It's a structured approach where team members come together to assign story points to backlog items. The process involves multiple estimation rounds facilitated by a Scrum Master, where each team member selects a Planning Poker card representing their estimate of the effort required. After revealing and discussing the chosen cards, a consensus estimate is reached and recorded for each backlog item. Specifically, into the project, the tool used was Chpokify, a tailored tool for doing planning poker in a remote way, an overview of the tool is in image 19.


SESSION

Product Team Estimation, Sprint 8.11, Part II

This is a description of the session.

TIME 00:25 SCORE 5

STORIES

Update empty states for stories and teams	0
Fix browser text fields autocomplete function on Sign in pages/modal windows	6
invite with email into the team	0
Update texts throw whole project and add extra descriptions	0
Onboarding design	3

CURRENT STORY

Update empty states for stories and teams

STORY SCORE 5

00:25

Front-end Product 5

udi	saitonakamura	rkhitin	connormiha	fixik
Pass	5	5	5	8

Back-end Product 0

disc, artos, pincet

Qa Product 0

nixdrey, photoelf, mariamskaya, dany

Figure 19: Chpokify Interface (https://chpokify.com)

As specified in the paper called “On using planning poker for estimating user stories” [15], *“discussion through planning poker can identify forgotten tasks and increase the initial estimates, the study has shown that the lowest and the highest estimates provided during the first round of planning poker act as strong anchors representing the lower and the upper boundary of the interval within which the group consensus is sought”*[1]. This is exactly what happened applying this methodology in a real-life project, my team have expert developers alongside recent graduates, leading to

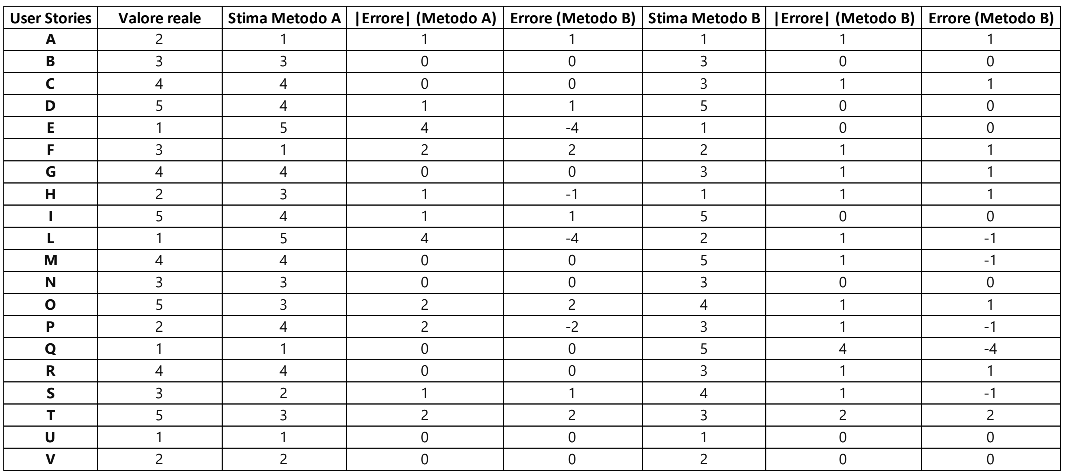
[1] On using planning poker for estimating user stories, Viljan Mahnic; https://www.sciencedirect.com/science/article

instances where the more experienced members asserted their ideas over those with less experience.

Effectiveness from data

To verify the validity of the previously stated claims, I conducted a small case study within the team to determine whether the Planning Poker method had indeed brought about a positive effect or not. This is especially crucial at a project level, as it is a highly delicate and critical subject, considering that Reply's compensation for the work performed is recognized based on the estimates provided to the client.

Twenty user stories were selected and estimated using both Method A (Scrum Master + Tech Leader) and Method B (Planning Poker).



User Stories	Valore reale	Stima Metodo A	|Errore| (Metodo A)	Errore (Metodo B)	Stima Metodo B	|Errore| (Metodo B)	Errore (Metodo B)
A	2	1	1	1	1	1	1
B	3	3	0	0	3	0	0
C	4	4	0	0	3	1	1
D	5	4	1	1	5	0	0
E	1	5	4	-4	1	0	0
F	3	1	2	2	2	1	1
G	4	4	0	0	3	1	1
H	2	3	1	-1	1	1	1
I	5	4	1	1	5	0	0
L	1	5	4	-4	2	1	-1
M	4	4	0	0	5	1	-1
N	3	3	0	0	3	0	0
O	5	3	2	2	4	1	1
P	2	4	2	-2	3	1	-1
Q	1	1	0	0	5	4	-4
R	4	4	0	0	3	1	1
S	3	2	1	1	4	1	-1
T	5	3	2	2	3	2	2
U	1	1	0	0	1	0	0
V	2	2	0	0	2	0	0

Figure 20: Estimation Table

Firstly, the following were calculated:

Absolute value error = | actual value - estimate Method A |

Error: actual value - estimate Method B

At a project level, what interested me most was that the sum of the errors did not deviate too much from zero. It's normal in the field not to have estimates that perfectly reflect reality; what's important is that there is some sort of compensation, for avoiding positive or negative trends.

As shown in the figure 20, using Method A, the summation of the error is -1, while using Method B, it's 1, indicating that in either case, the two methods can be considered valid and equivalent. While, observing the average error, there is a reduction when using Method B, which tends to suggest that Method B is the more accurate and precise method, although it is not such a substantial difference.

AVG Errore Metodo A	1,05
AVG Errore Metodo B	0,85
Somma errori metodo A	-1
Somma errori metodo B	1

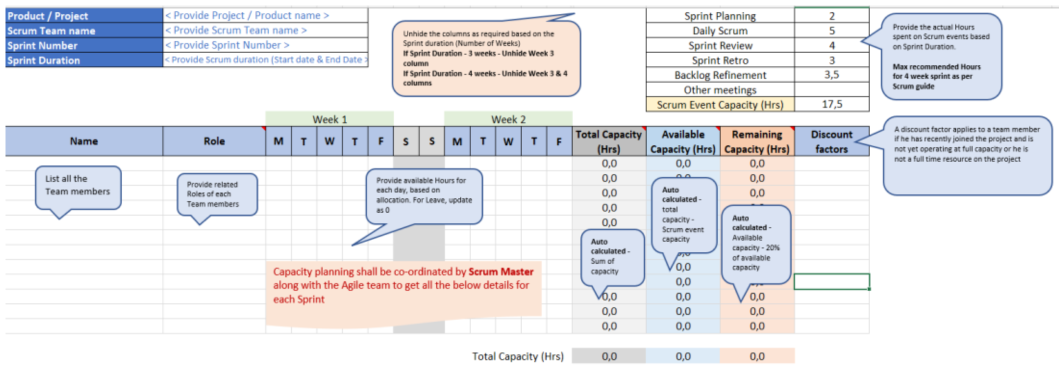
The last point not covered by the data is the spread of knowledge within the team and the increased awareness of younger team members. During the dedicated Planning Poker meetings, there is an opportunity to discuss, learn, and enhance one's knowledge, both technically and in terms of work.

5.2 Capacity Planning

One of the critical activities of the Scrum Master role involves planning the PI (Product Increment). Capacity planning assists the team in estimating the necessary capacity to accomplish the work in the upcoming sprint. It explains the amount of work a team can undertake without compromising quality, productivity, and efficiency. Planning facilitates the balancing of team members' available hours with the project's demands. Put simply, in Agile, capacity refers to the volume of work achievable within a given timeframe, in this specific case one PI.

Prior to the start of the PI, the capacity must be calculated for its entire duration (2 months), encompassing the subsequent 4 sprints. The total workload that can be loaded into a PI is determined by the sum of the capacities of the 4 sprints.

To calculate the capacity, I've devised the following template illustrating the plannable capacity within a single sprint.



Product / Project	< Provide Project / Product name >
Scrum Team name	< Provide Scrum Team name >
Sprint Number	< Provide Sprint Number >
Sprint Duration	< Provide Scrum duration (Start date & End Date >

Unhide the columns as required based on the Sprint duration (Number of Weeks)
If Sprint Duration - 3 weeks - Unhide Week 3 column
If Sprint Duration - 4 weeks - Unhide Week 3 & 4 columns

Sprint Planning	2
Daily Scrum	5
Sprint Review	4
Sprint Retro	3
Backlog Refinement	3,5
Other meetings	
Scrum Event Capacity (Hrs)	17,5

Provide the actual Hours spent on Scrum events based on Sprint Duration.

Max recommended Hours for 4 week sprint as per Scrum guide

		Week 1							Week 2								
Name	**Role**	**M**	**T**	**W**	**T**	**F**	**S**	**S**	**M**	**T**	**W**	**T**	**F**	**Total Capacity (Hrs)**	**Available Capacity (Hrs)**	**Remaining Capacity (Hrs)**	**Discount factors**
														0,0	0,0	0,0	
														0,0		0,0	
														0,0		0,0	
														0,0			
														0,0			
															0,0		
															0,0		
														0,0	0,0	0,0	
														0,0	0,0	0,0	
														0,0	0,0	0,0	
													Total Capacity (Hrs)	0,0	0,0	0,0	

List all the Team members

Provide related Roles of each Team members

Provide available Hours for each day, based on allocation. For Leave, update as 0

Capacity planning shall be co-ordinated by **Scrum Master** along with the Agile team to get all the below details for each Sprint

Auto calculated - Sum of capacity

Auto calculated - total capacity - Scrum event capacity

Auto calculated - Available capacity - 20% of available capacity

A discount factor applies to a team member if he has recently joined the project and is not yet operating at full capacity or he is not a full time resource on the project

Figure 21: Template for calculating Capacity

For each team member, the available working hours for each day of the week are recorded to account for vacation days, leaves, or other absences. Once the total capacity for each team member is calculated, it is reduced by subtracting the hours allocated to Scrum ceremonies and other meetings scheduled for the upcoming sprint. The result of this operation is referred to as the available capacity.

The final step to determine the capacity effectively allocated to user stories is to reduce the available capacity by 20%. After multiple iterations and observing the collected data on velocity (further explored in KPI), it was found that a 20% reduction is the correct discount factor to apply for non-backlog items.

Non-backlog items include:

- Defects: Possible bugs in the code after product release in E2E.
- Incidents: Bugs occurring in production, requiring immediate and resolutive intervention.
- Grooming: Analysis of epics to be implemented in the next PI.

Lastly, there is a column for the discount factor, which reduces the capacity of team members who are new to the project. The onboarding path for a new member who has become fully integrated after a year in the project is as follows: the first month is dedicated entirely to knowledge transfer (KT), which involves studying documentation and video lessons to understand the product and work systems. Consequently, the resource's capacity will be set to 0. In the first sprint following the KT month, a discount factor of 75% is applied. In the second sprint in which the resource actively works on the project, their discount factor decreases to 50%. From the third sprint onwards, they can be considered full-time resources, with efforts made to assign tasks that are not overly complex and do not require an in-depth knowledge of the product. This process has been established because over the course of a year, the team has undergone several changes, transitioning from a total of 5 members to the current 12 members.

5.2.1 How to tackle planning fails

Having experienced various changes within the team over the past year, planning cannot always be perfect. Therefore, it's essential to implement preventive measures to react immediately and ensure the project's safety.

Two situations may arise: underestimating capacity, meaning too few story points are loaded into the sprint backlog, or overloading with too many story points, jeopardizing delivery commitment and risking the inability to deliver all epics committed to during PI planning.

The first scenario is relatively easier to manage, with the most readily adoptable solution being to start working on user stories planned for the next sprint, i.e., including stories from the next sprint into the current sprint backlog. If this solution isn't feasible due to dependencies between epics or external teams, as suggested by Scrum methodology, it's important to always have refined user stories in the backlog. In this case, an overall backlog contains user stories and epics not planned for the current PI. These stories can then be directly inserted into the current sprint backlog.

The second scenario is the more complicated one and requires precise timing of intervention to avoid impacting the project's progress. Since a team represents a sub-product, if it fails to deliver what it has set out to, the entire product can be affected by it. Therefore, it's essential to address overloading or capacity issues promptly and effectively to mitigate any negative repercussions on the overall project. In Agile development, "spillovers" [16] typically refer to epics or work items that are not completed within the planned timeframe, such as a sprint. These unfinished tasks spill over into subsequent iterations or sprints, impacting the overall project timeline and potentially causing delays. The process begins during sprint planning, where the team identify the most critical tasks. Throughout the sprint, continuous monitoring is conducted. As soon as it becomes apparent that there isn't sufficient capacity to complete all items in the backlog, it's essential to prioritize the most critical tasks. Additionally, transparent communication with the client regarding the at-risk activities is imperative.

5.2.2 Possible causes of spillover

There are several potential causes behind spillovers, and not all of them are applicable to every project. Below is a list of the most frequent ones accountable to the Product Owner which I have observed over the past year and a half of the project:

- Changing requirements: when the Product Owner offers unclear or frequently changing requirements within a sprint, it can lead to misunderstandings or frequent adjustments into the ongoing sprint, causing tasks to overflow into subsequent sprints.

- Lack of prioritization: if the Product Owner fails to prioritize tasks effectively or lacks awareness of various prioritization techniques, the team may focus on lower priority tasks initially, neglecting high-priority stories or tasks by the end of the sprint.

- Lack of Availability for Clarifications: when the Product Owner is inaccessible for clarifications or feedback during the sprint, it can delay decision-making, impede the team's progress, and result in incomplete stories.

- Ambiguous Acceptance Criteria: unclear or poorly defined acceptance criteria of epic set by the Product Owner can lead to incomplete or erroneous implementation, necessitating rework and causing tasks to overflow. This cause is one of the most observed in the project.

- Sprint Backlog Overload: providing an excessive workload or continuously adding tasks during a sprint without considering the team's capacity can overwhelm the team, resulting in unfinished tasks.

It's crucial to note that in the described project, the Product Owner is part of the client company. Consequently, it's the responsibility of the Scrum Master to document any potential spillovers caused by the above mentioned factors and promptly communicate them to the Product Owner. This approach allows for a constructive discussion aimed at improving the project process. As an immediate action, it may be beneficial to re-estimate some stories which will spill, that were already part of the Sprint Backlog.

While some of the causes are attributable to the Scrum Master and the development team and will now be described:

- Poor Sprint Planning: if the Scrum Master fails to assist in planning sprints effectively, the team may struggle to understand or organize tasks correctly. This could lead to underestimation of work or ineffective task organization, resulting in stories spilling over.

- Inadequate Communication: facilitating communication within the team is a key responsibility of the Scrum Master. Ineffective communication or unclear directives may cause misunderstandings, delays in issue resolution, or insufficient guidance, ultimately leading to stories carrying over to subsequent sprints.

- Inability to Address Impediments: a crucial role of the Scrum Master is to identify and remove obstacles hindering the team's progress. If they are ineffective in promptly addressing impediments, it can result in task completion delays and spillover stories.

- Insufficient Support to the Team: if the Scrum Master fails to provide adequate assistance or guidance, the team may struggle to collaborate effectively or resolve problems efficiently. This may result in incomplete stories by the end of the sprint.

- Neglecting Retrospective Facilitation: Retrospectives play a vital role in continuous improvement. If the Scrum Master does not conduct effective retrospectives or fails to follow up on action items, it may lead to recurring issues, causing stories to spill over from one sprint to another. The retrospective ceremony will be described in detail in the last chapter.

- Underestimation: sometimes, insufficiently thorough grooming can lead to the underestimation of certain tasks. Consequently, the user stories in the sprint backlog may include unforeseen work that needs to be completed.

- Unplanned Leaves: it's essential for the Scrum Master to plan the team's capacity for the Program Increment (PI) considering the vacations of each team member. An unplanned leave, whether due to uncommunicated holidays or absence days caused by illness, can result in a reduced team capacity compared to what was communicated at the beginning of the PI. Consequently, this can lead to spillovers.

- Defects Requiring Rework: only 80% of the total capacity is allocated to the development of user stories, with the remaining 20% reserved for addressing defects. If the product released by the team is of poor quality and incurs a higher number of defects than anticipated, this will result in a decreased allocable capacity for user stories. Defects take precedence over future developments, thereby reducing the available capacity for user stories.

- No Active Participation in Scrum Ceremonies: during Scrum Ceremonies, issues and potential solutions are discussed. If team members are not actively participating or are distracted, this can lead to blockers and delays in development, resulting in spillovers.

These are some of the possible causes identified during the project. It's essential to intervene immediately to prevent spillovers from becoming a habit and having a detrimental impact on the project timeline.

5.3 KPI

Key Performance Indicators (KPIs) in Agile development are metrics used to measure the effectiveness, efficiency, and performance of Agile teams and projects. These indicators provide valuable insights into various aspects of the Agile process, allowing teams to assess their progress, identify areas for improvement, and make data-driven decisions.

As the Scrum Master of the team, I am responsible for continuously monitoring, calculating, and interpreting the main key performance indicators (KPIs) [17]. This involves regularly tracking metrics such as velocity, focus factor percentage of found work, percentage of adopted work [18], defects received in each sprint and team happiness index, and analyzing the data to identify trends, patterns, and areas for improvement. I communicate the findings and insights to the team, stakeholders, and other relevant parties, facilitating discussions and collaborative problem-solving sessions to address issues and obstacles. By actively monitoring and interpreting the KPIs, I ensure that the team remains focused on its goals, identifies areas for improvement, and continuously adapts its approach to deliver value effectively. Anyway, there are common KPIs that many Agile teams use, but the important KPIs for an Agile project can vary depending on the project context and the specific goals and challenges of the team. In the following paragraph are described the main important KPI for the analyzed project.

- Velocity is one of the main KPIs for an Agile team it represents the amount of work completed by the team during a sprint, typically measured in story points or other units of work. In this specific case, velocity is indeed the most important KPI, serving a crucial dual purpose both within the team and externally for reporting results to Reply's management. Internally, velocity allows for the evaluation of team performance, helping to predict future sprint capacity and providing insights into team efficiency. Externally, project revenue relies solely on the quantity of delivered story points, making velocity highly monitored by management. While looking solely at velocity data may not capture all project dynamics, analyzing past sprints allows for the establishment of a baseline for predicting future project revenues and initiating potential analyses in the event of underperformance or overperformance by the team. An efficient tool for monitoring the velocity during the sprint is the burndown chart (image 22);

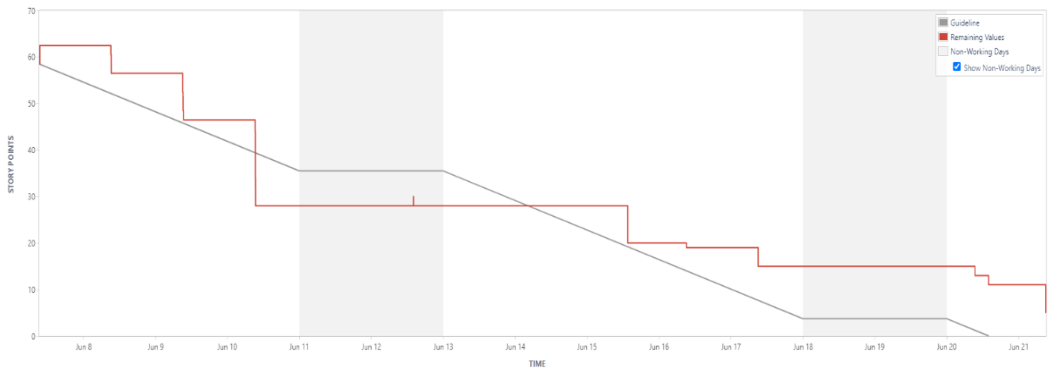


Figure 22: Burndown Chart

- Another important KPI that I found in literature is the Focus Factor, calculated as the ratio of Velocity to Work capacity, ideally is around 80% on average for a healthy team. If the is team facing challenges during the sprint, this indicator should be below 80%. The interpretation of this KPI below 80% suggests disruptions from external events or difficulties in converting forecasted work into accepted work. Conversely, if the Focus Factor becomes too high, it often indicates that the team may have underestimated their capabilities to maintain an illusion of perfection or overlooked other organizational responsibilities, which could lead to future issues.

- *Percentage of Found work* is calculated as:

$$\sum(Original\ Estimates\ of\ Found\ Work) \\ /(Original\ Forecast\ for\ the\ Sprint) * 100\%$$

The term Found Work refers to additional work discovered once work on the stories has already started, measured in story points. For example, if the additional work requires 3 extra workdays, the found work would be equal to 3. By dividing the number of story points added by the total number of story points initially planned for the analyzed epic, the percentage of work that was not initially planned is determined. This KPI can provide efficient insights into the quality of the team's previous analysis. It is crucial because after further analysis of the epic that required additional work, it can be determined whether it was a shortfall on the part of the team and therefore an analysis done with poor attention, or if the additional work is attributable to the client due to incorrect drafting of the epic's requirements. If the latter scenario arises, it is necessary to schedule a meeting with the client to explain the situation and request additional compensation beyond what was initially planned.

- *Percentage of Adopted Work:*

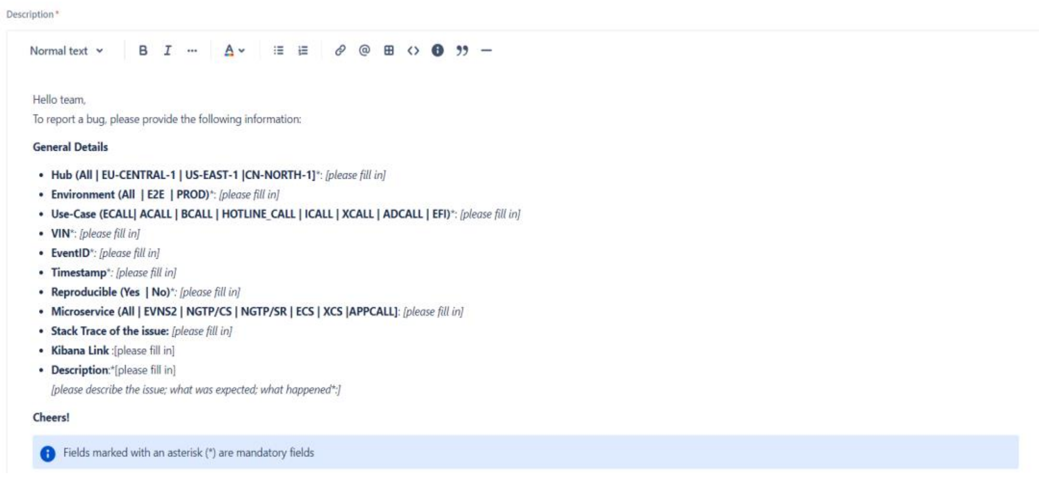
$$\sum(Original\ Estimates\ of\ Adopted\ Work) \\ \div\ (Original\ Forecast\ for\ the\ Sprint) * 100\%$$

Adopted work refers to user stories from the Product Backlog that are pulled into the current Sprint if the team completes their original forecast ahead of schedule, it is calculated as amount story points. A value exceeding 20% in this KPI opens up potential interpretation scenarios: it could be caused by poor planning on the part of the Scrum Master, indicate an improvement in the team that has increased velocity through corrective actions taken, or as a last hypothesis, it could be due to the fact that the user stories already present in the sprint backlog were overestimated. As in the previous scenario, if the latter case is identified, a meeting with the client would be necessary. Being transparent with the client even in these cases where there could be an advantage, it helps build a relationship of trust with the client, leading to significant benefits for the team and the project itself.

- *Defects (bugs) received in each sprint*:

$$\sum Defects\ in\ each\ sprint$$

Defects or bugs are potential code imperfections encountered when the product is released in end-to-end environment and testing phase is started. Since the beginning of the project, I have been tracking the quantity of defects received in each sprint for several fundamental reasons. Firstly, it provides an easily interpretable means to monitor the quality of the delivered product. After a few sprints, establishing a baseline allows us to gauge whether there has been an improvement or deterioration in the quality of the delivered code. Secondly, this metric serves as a measure of customer satisfaction. Beyond the trust built with the client, having few defects in the released product ensures a smoother workflow for the team and avoids the costs of rework. Thirdly, delving deeper into the analysis behind the raw data, I noticed that only 15% of the defects assigned to my team were actually attributable to the code implemented by my team. This necessitated a thorough analysis of each ticket by the team before redirecting it to the relevant team, thus consuming time that could have been allocated to the development of user stories. By demonstrating to the client that there was an issue with defect rerouting, we were able to adopt a new template (image 23) to provide to testers, making it easier to assign tickets to the correct team.



Hello team,
To report a bug, please provide the following information:

General Details

- **Hub (All | EU-CENTRAL-1 | US-EAST-1 |CN-NORTH-1]***: *[please fill in]*
- **Environment (All | E2E | PROD)***: *[please fill in]*
- **Use-Case (ECALL| ACALL | BCALL | HOTLINE_CALL | ICALL | XCALL | ADCALL | EFI)***: *[please fill in]*
- **VIN***: *[please fill in]*
- **EventID***: *[please fill in]*
- **Timestamp***: *[please fill in]*
- **Reproducible (Yes | No)***: *[please fill in]*
- **Microservice (All | EVNS2 | NGTP/CS | NGTP/SR | ECS | XCS |APPCALL]**: *[please fill in]*
- **Stack Trace of the issue:** *[please fill in]*
- **Kibana Link** :[please fill in]
- **Description**:*[please fill in]
  [please describe the issue; what was expected; what happened:]*

Cheers!

Fields marked with an asterisk (*) are mandatory fields

Figure 23: Defect Template

- Last but not least, *Happiness of your team!*

  It’s the more subjective metric but is one of the most important. Team happiness in Agile refers to the overall satisfaction and morale of the team members working within an Agile framework. It encompasses various aspects such as the team's sense of accomplishment, work-life balance, job satisfaction, and alignment with project goals and values.
  As a Scrum Master, it's important to be empathetic with team members and try to understand any potential discontents or difficult personal moments some team members may be experiencing. However, it's not always easy to gauge the situation accurately, which is why some actions can be taken to uncover the real situation.
  In a project managed entirely remotely like this one, every meeting becomes crucial for understanding the team's mood, but the most important meeting is undoubtedly the retrospective, a topic that will be addressed in the next paragraph.

5.4 Integration of the risk management process in Scrum

Studying the Scrum methodology and applying it to the project, I noticed that the risk management aspect was not formally addressed, but rather managed intrinsically as the project progressed. In searching the literature, I looked into how to best implement the risk management process in Scrum. The main method I found include RBSM (Risk Based Scrum Method) [19], in which, teams focus on identifying potential risks early in the project and continuously throughout its lifecycle. These risks are then analyzed, prioritized, and addressed using appropriate mitigation strategies. By integrating risk management into the Scrum framework, RBSM aims to enhance project success by minimizing the impact of potential threats and maximizing opportunities for project advancement. RBSM provides a structured approach to risk identification, assessment, and response, ensuring that risks are effectively managed while maintaining the iterative and adaptive nature of Scrum. Anyway, RBSM lacks mechanisms for incorporating new risks that arise from evolving requirements and does not provide a defined process for recording these risks along with their mitigation plans.

5.4.1 Process adopted

The process proposed in the thesis was initially identified in the existing literature [20] and then customized to fit the specific requirements and characteristics of the analyzed project.

The core risk management activities include risk identification, analysis, mitigation, and control. Given that Scrum is inherently iterative, the method proposes that the risk management process should also follow an iterative approach. For risk identification, we utilize brainstorming sessions where the Scrum team collectively identifies risks based on their past project experiences or any anticipated technical challenges. During each sprint planning session, a brainstorming session occurs where the team discusses every selected user story from the backlog. The aim is to identify potential risks and dependencies that could jeopardize the sprint goal. All the risks identified before are documented in a risk register. Subsequently, risk assessment takes place during a dedicated risk meeting (once a week), where the team conducts risk analysis. Factors such as risk impact and probability are determined based on past experiences and consultation with a risk repository containing similar project risks. Finally, a risk mitigation plan is devised for each identified risk. This iterative risk management process is integrated with the Scrum framework, ensuring alignment with project development cycles as it is visible in the image below.

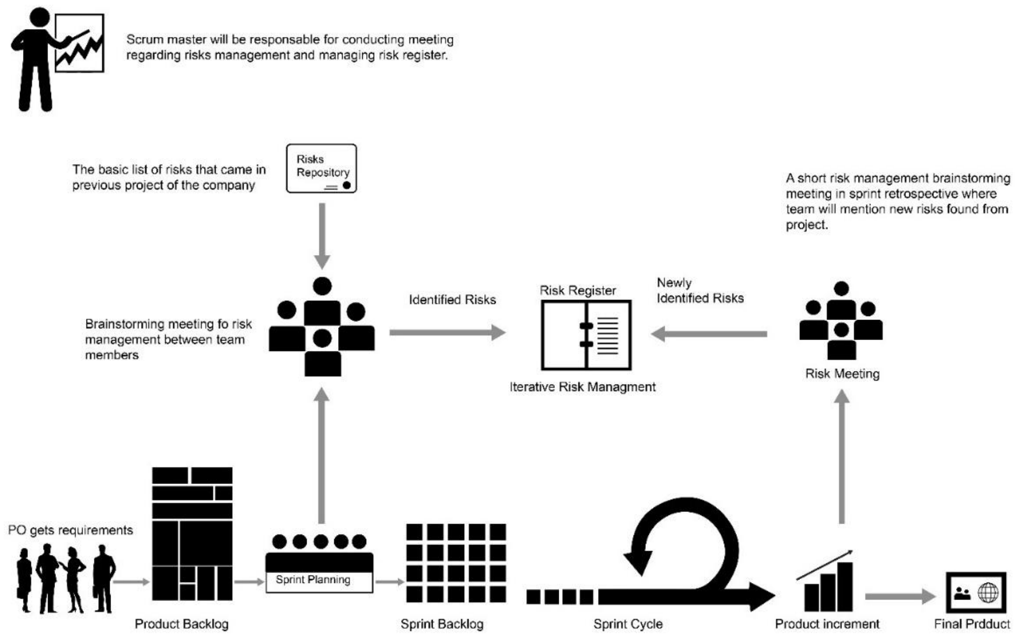
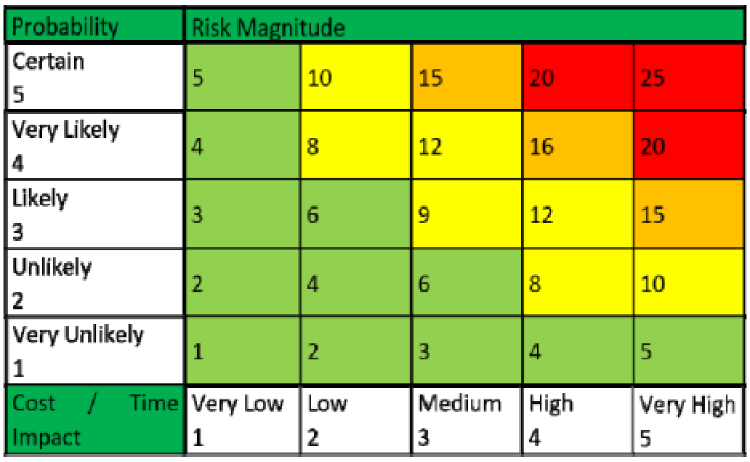

Scrum master will be responsable for conducting meeting regarding risks management and managing risk register.
The basic list of risks that came in previous project of the company
Risks Repository
A short risk management brainstorming meeting in sprint retrospective where team will mention new risks found from project.
Brainstorming meeting fo risk management between team members
Identified Risks
Risk Register
Newly Identified Risks
Iterative Risk Managment
Risk Meeting
PO gets requirements
Product Backlog
Sprint Planning
Sprint Backlog
Sprint Cycle
Product increment
Final Prdduct

Figure 24: Integration of Risk Management in Scrum Framework [20]

The model found in the literature described a project managed entirely internally, where the Product Owner (PO) served as the risk manager. However, in the case of a consultancy project, the Scrum Master assumes the role of the risk management process owner. So the Scrum Master, is accountable for monitoring, taking care of risks and risk register and of the communication with the stakeholders.

Then, after the first analysis is done, with the risk matrix is possible to assert the risk magnitude of each risk identified. The risk magnitude is a value that is found by multiplying the cost associated with a given risk coming true by the probability of the given risk happening. Both variables have values that lie on a scale from 1 to 5, with 1 being the minimum probability of risk occurrence and 5 being the maximum.

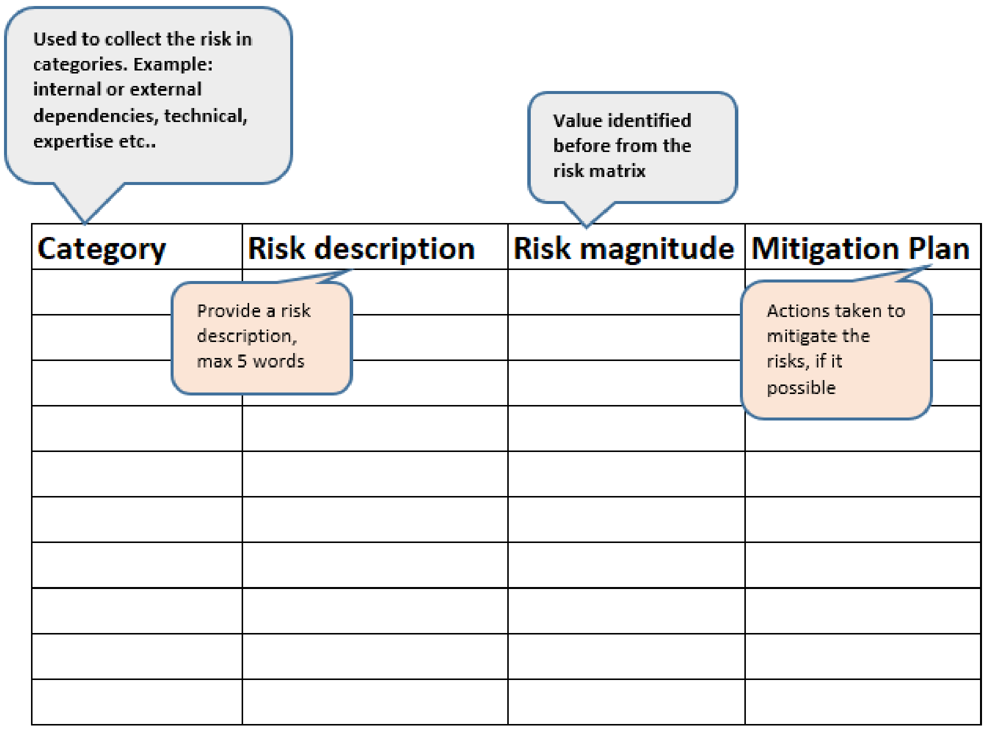


Probability	Risk Magnitude				
Certain 5	5	10	15	20	25
Very Likely 4	4	8	12	16	20
Likely 3	3	6	9	12	15
Unlikely 2	2	4	6	8	10
Very Unlikely 1	1	2	3	4	5
Cost / Time Impact	Very Low 1	Low 2	Medium 3	High 4	Very High 5

Figure 25: Risk Rating Matrix [20]

Once the risk has been identified and evaluated, it is recorded in two different risk registers. The first is the risk register for the current sprint, while the other risk register serves as a database, allowing it to be consulted whenever a previously encountered risk needs to be addressed. The template is shown in image 26.

After adopting this process for an entire Product Increment (PI), the benefits became immediately apparent. Not only was risk managed, where possible, with actions aimed at preventing it, but it also proved to be an excellent ally during planning. Sometimes there are risks that cannot be mitigated, so by analyzing the risk register of the sprint and summing the "Risk Magnitude" column, it is possible to make adjustments to the planning if the calculated value is excessive, thus avoiding overly impacting the team's performance if all possible risks were to materialize.



Category	Risk description	Risk magnitude	Mitigation Plan

Used to collect the risk in categories. Example: internal or external dependencies, technical, expertise etc..
Value identified before from the risk matrix
Provide a risk description, max 5 words
Actions taken to mitigate the risks, if it possible

Figure 26: Risk Register

5.5 Retrospective

Starting with a quote from Steve Jobs, "*Sometimes when you innovate, you make mistakes. It is best to admit them quickly, and get on with improving your other innovations.*[2]" [21] This sentiment underscores the significance of retrospective meetings. Even the most proficient product teams encounter mistakes, whether due to oversights, disorganization and misunderstandings.

The key is to recognize that making mistakes is acceptable it's the learning from them that matters. Failure to learn can lead to recurrent issues with potentially significant consequences, affecting not only the team but also end-users. This is where agile retrospectives come into play. They gather the product development team to reflect on the process, assess what worked and what didn't, and commit to improvements for the future. While the primary focus in an agile environment might be on swiftly delivering a product to customers, it's essential for teams to circle back and evaluate their work.

Based on my experience and studies in literature, it is evident how the retrospective, a Scrum meeting, is often underestimated and not given in real consideration by the team. From the experience gained on the project, I can affirm how it is instead one of the most important Scrum Ceremonies, but at the same time the most delicate due to the role of the Scrum Master, who must be able to enable everyone to express themselves freely. Therefore, with a full remote working mode, it is important to find easily understandable games that can keep the team's interest alive. In the next paragraph, I will explain the main games used in the project. The Scrum methodology dictates a standardization in project execution, which implies that the team must work closely together continuously. Therefore, it is necessary to understand what is going wrong to avoid continuing to have such situations, and what is going well to continue along that path. Additionally, a predisposition to continuous improvement is intrinsic to Agile methodologies, which is why the retrospective perfectly embodies one of the fundamental principles of Agile.

[2] As reported in Retrospective, https://airfocus.com

The retrospective ceremony takes place after the completion of each sprint, and after witnessing the benefits brought by the retrospective to the team, I have decided to propose a second retrospective, but this time directly involving the client at the end of each Program Increment (PI). This second retrospective has not only improved the daily work processes with the client, but it has also brought significant benefits to the trust relationship that was being established with the client.

5.5.1 Internal Retrospective

To keep the team engaged during our meetings, I've made an effort to vary the template and activity used since the beginning of the year. Additionally, I've implemented small changes to prevent the team from becoming complacent and to emphasize the significance of the ceremony. The tool used for the retrospective was Mural, an online collaboration platform that allows users to create, share, and collaborate on virtual whiteboards in real-time. It is designed to facilitate teamwork, enabling users to draw, write, add sticky notes, and organize visual content intuitively. Mural is widely used in business, education, and creative fields for planning, brainstorming, project management, and other collaborative activities.

Game A

The first template used is visible in the image. Each team member has a box where they can express their ideas by writing on colored sticky notes.

The colors represent three different categories:

- Green indicates what went well during the sprint, or what needs to be continued to maintain the desired level of quality and process. It may be helpful in this section to highlight actions taken by a specific team member so that it can serve as an example for others.

- The purple section represents what did not go well, or what should not be repeated to avoid harm to the project. It is probably the most delicate and important section of the retrospective, and it is usually where the most heated discussions take place. The Scrum Master must be skilled in moderating tones and conveying the importance of this part of the ceremony; it should be seen as a moment of personal and team growth.

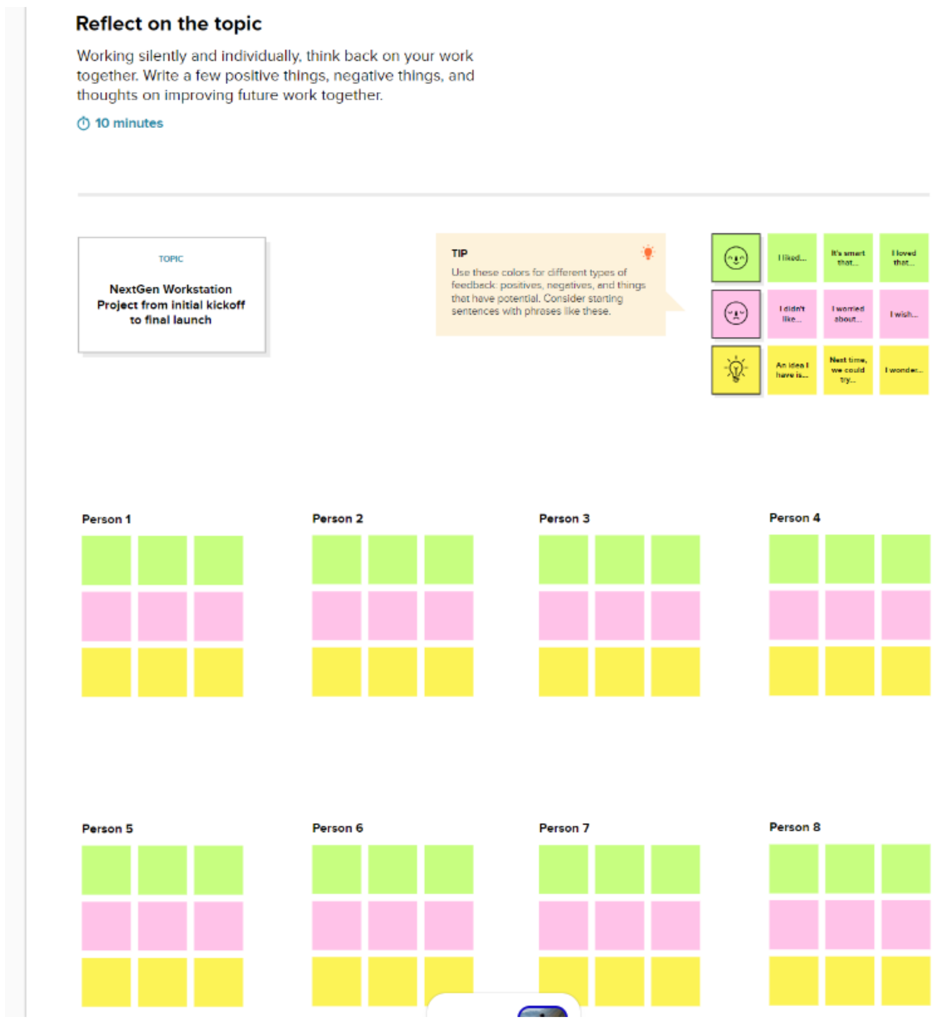
- The yellow category represents ideas that can benefit the project. It is usually discussed after the purple category, to have new ideas that can have an improving effect on what was previously discussed.

  The meeting process is quite simple. Usually, in the first 10 minutes, the Scrum Master presents a brief overview analyzing improvement actions (the yellow category) discussed in the previous retrospective. This helps to highlight the true value of the meeting, providing the team with evidence of past discussions. Moreover, it helps to assess whether implemented ideas have had a positive effect, allowing for their continued application or refinement if they impacted another process and need adjustment.

  Following this, the discussion moves to the purple category, focusing on what didn't go well in the previous sprint. Using Mural, a timer can be set, typically allowing 3 to 4 minutes for each team member to write their thoughts on sticky notes. Once time is up, each team member takes turns to comment on what they've written, sparking a brief discussion. The role of the Scrum Master here is crucial; they must prevent discussions from becoming overly prolonged and ensure that every team member can express their ideas calmly, preventing one person from dominating.

  After addressing what went wrong, the team discusses new implementation ideas aimed at improving the project process. Often, inspiration is drawn from what didn't go well, sparking fresh ideas and considerations for application.

  Finally, the green section is discussed, highlighting what went well and what should be continued. While this section may seem less important, it holds equal importance to the others in ensuring a comprehensive retrospective.


Reflect on the topic

Working silently and individually, think back on your work together. Write a few positive things, negative things, and thoughts on improving future work together.

10 minutes

TOPIC

NextGen Workstation Project from initial kickoff to final launch

TIP

Use these colors for different types of feedback: positives, negatives, and things that have potential. Consider starting sentences with phrases like these.

Figure 27: Template Game A

Game B

Game B is very similar to Game A, with the layout change designed to highlight the shift in process. The categories can be considered the same as the previous game, with the only difference being that the category of new improvement ideas, are in the "Actions" column, marked by blue sticky notes.

In Game A, each sticky note was written during the meeting. Observing that meetings often focused mainly on topics from the second week of the sprint, I proposed a process change: sticky notes could be created from the first to the last day of the sprint, not just during the retrospective ceremony.

With this approach, once the meeting began, we started by analyzing and discussing the existing sticky notes. Once the initial phase was concluded, the same process as described in Game A would commence. It was a slight process change but resulted in significant benefits for the team. By doing so, every aspect of the sprint could be considered and not forgotten.

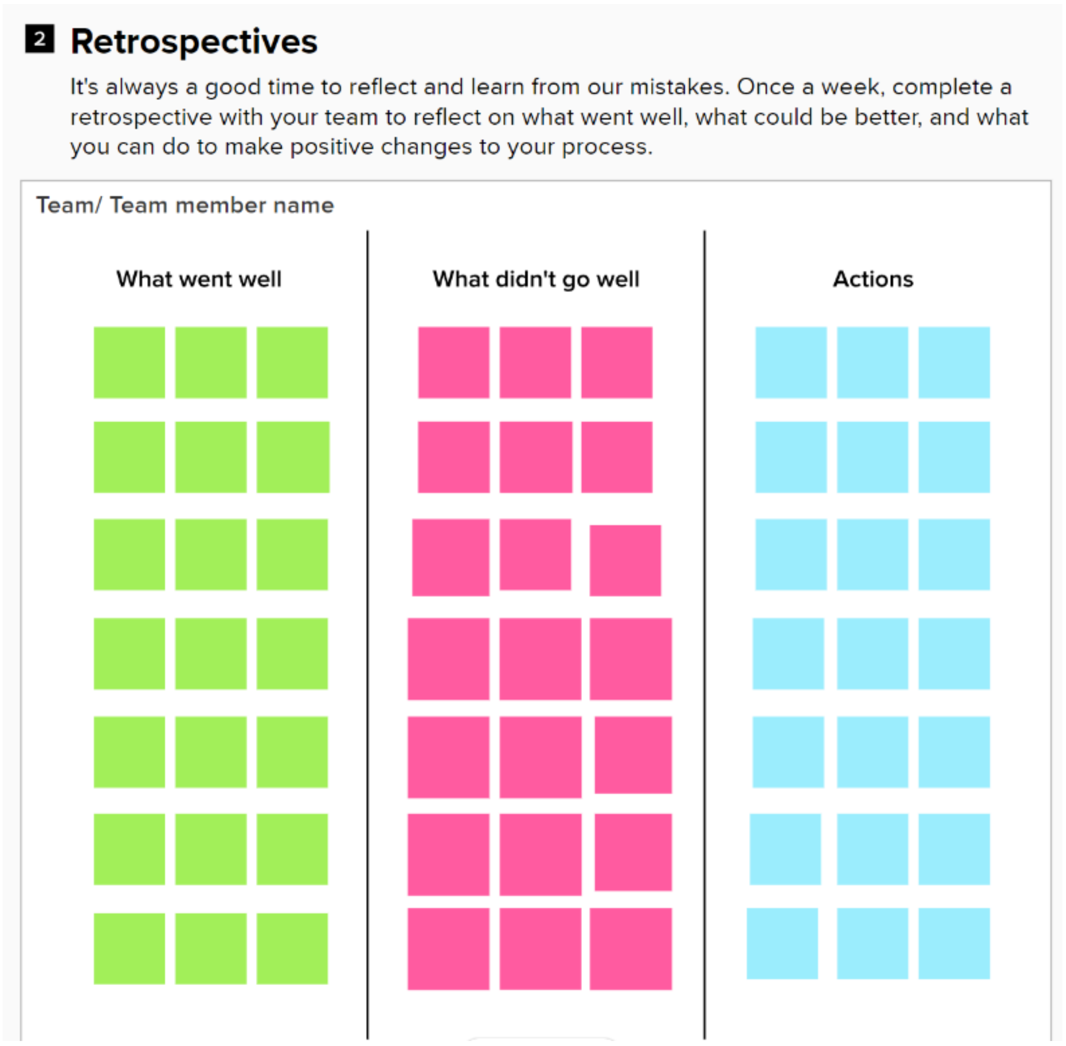

2 Retrospectives
It's always a good time to reflect and learn from our mistakes. Once a week, complete a retrospective with your team to reflect on what went well, what could be better, and what you can do to make positive changes to your process.
Team/ Team member name
What went well
What didn't go well
Actions

Figure 28: Game B Template

Game C

In the project the team currently employs the Starfish game model, a framework sourced from literature that incorporates two additional categories alongside the typical three.

The Starfish game represents an innovative approach to the traditional retrospective process. Its game board features 5 different columns, each with a different focus:

- Stop Doing: This area addresses activities or practices that haven't yielded value or have even impeded progress.

- Less Of: Here, the team considers activities or practices that have added value but required excessive effort.

- Keep Doing": This segment celebrates activities or practices that the team performs well and wishes to maintain.

- More Of: Activities or practices deemed useful but underutilized are highlighted here, with the team recognizing potential for increased value if given more attention.

- Start Doing: This section prompts the team to introduce new activities or practices they believe will contribute positively.

Regarding the process, it is similar to that described in Game B, where each team member can create sticky notes throughout the sprint, and these will be the first items discussed at the beginning of the ceremony. Subsequently, there will be a few minutes to add additional sticky notes for discussion. The modification, compared to what is described in Game B, is solely that there is no turn-taking where each team member presents what they have written, thus initiating the discussion. In this case, after everyone has finished placing the sticky notes in the category under consideration, the Scrum Master groups the sticky notes by topic.

By doing so, the discussion will have specific topics, ensuring that all ideas expressed by each team member are considered for that particular subject.

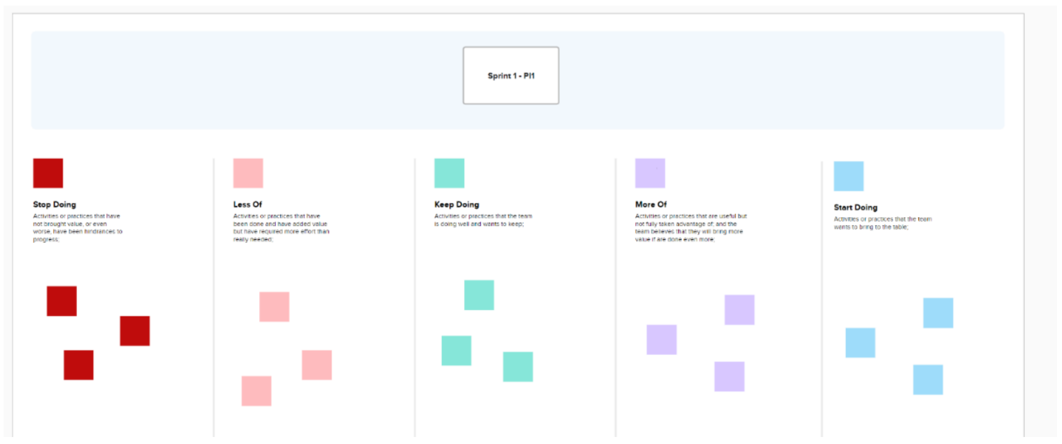

Sprint 1 - PI1
Stop Doing
Activities or practices that have not brought value, or even worse, have been hindrances to progress;
Less Of
Activities or practices that have been done and have added value but have required more effort than really needed;
Keep Doing
Activities or practices that the team is doing well and wants to keep;
More Of
Activities or practices that are useful but not fully taken advantage of; and the team believes that they will bring more value if are done even more;
Start Doing
Activities or practices that the team wants to bring to the table;

Figure 29: Game C Template

6 Conclusion

The thesis aimed to conduct an analytical and critical study on how a consulting company as Reply applies Agile methodology to a project that is currently live in production.

In the first part of the study, an overview of the main project management methodologies was provided, starting with an analysis of the waterfall methodology and its key frameworks. Subsequently, the paradigm shift with the adoption of Agile methodologies and its frameworks was explained.

The two frameworks examined were SAFe and Scrum, two distinct frameworks that can coexist harmoniously within the same project. To emphasize this, introductions to both frameworks were provided, followed by an analysis of their core values and operational methods. This analysis aimed to highlight the differences, as SAFe falls under the category of methods for scaling Agile, i.e., applying Agile methodology at a product level with multiple teams using the Scrum framework at a sub-product level.

In the second part, following an initial introduction of the company Reply it’s provided an overview of the project's field of application, the context in which the project operates is described, called the world of Connected Cars. Subsequently, the project scope is outlined, including an analysis of the involved teams, to underscore the differing application of the SAFe and Scrum methodologies. This is followed by a description of the project timeline and a high-level analysis of project processes, providing the necessary insights to understand the project timeline.

This resulted in the focal point of the study, where a project undertaken by the company was closely observed from within. Consequently, the study focuses on the application of Agile methodology to a real project and the problems and challenges encountered during its implementation. Moreover, it highlights the most important and critical processes and how action was taken to improve them.

All actions taken to enhance the team and project progress are initially analyzed at a theoretical level, before being actually applied and their effectiveness measured through data.

6.1 Benefits

The main benefits brought by the study include proposing a guideline for applying Agile methodology to a real project and emphasizing the process differences from theory to practice. This paper can be utilized in the future by Reply as it encompasses various processes and improvements tailored to Agile methodology.

Several benefits have been identified, starting with one of the most crucial processes of Agile methodology, the Backlog Refinement. The detailed analysis conducted prior to implementation, involving the creation of a project document as depicted in the figure 16, resulted in reduced rework costs. This was achieved by thoroughly describing each user story related to an epic and presenting it to the client for approval before the team starts actual work on the user stories. Additionally, it facilitated the resolution of various blockers before implementation began, thereby facilitating the team's work planning and matching the predefined delivery deadline.

Staying within the area of Backlog Refinement, after initially estimating user stories solely with the ownership of the tech leader and the scrum master, integration of the Planning Poker process was made possible thanks to literature review findings. This technique not only enabled the provision of more accurate estimates to the client but also solidified the hard skills of younger team members, empowering them in one of the most delicate processes within the consultancy industry.

Moving forward, the thesis delves into tackling one of the most challenging aspects for a Scrum Master: planning. Starting from the provided template as depicted in the figure 21, it becomes possible to calculate the team's capacity within a single sprint while applying the discount factors discussed earlier. It is important to emphasize that the discount factors applied to capacity stem from months-long analyses and are consolidated within this project; they cannot be universally applied to other projects without conducting a deep analysis.

Despite efforts to anticipate and calculate every detail, unforeseen circumstances may still arise. Therefore, the paper presents few worst-case scenarios and offers insights on how to react in case the predetermined workload cannot be fully completed on time.

To monitor processes and ensure that new application solutions indeed have a positive impact on project performance, it is imperative to measure with actual data. That's why the thesis focuses on Key Performance Indicators

(KPIs). The research effort has led to the identification of the six most critical KPIs, which allow not only the monitoring of project progress during each sprint but also form a small database. This database provides an overview to support strategic decision-making, such as team composition.

Compared to older project management methodologies, Agile typically doesn't prioritize risk management. In this regard, the paper suggests integrating a risk management process into Scrum. The benefits of this integration have been many. It not only enabled the team to identify even trivial risks that could have jeopardized meeting predetermined deadlines but also assisted the Scrum Master in analyzing and prioritizing tasks in case a risk couldn't be avoided.

After two product increments (PIs) during which the process was utilized, it was then extended to the final client. This significantly improved client relationships, as the process was scrutinized in every detail, and the delivery improvements were evident. After two months of implementation, the team's performance continued to enhance, with a clear improvement in terms velocity of the team, following the integration of the process.

The last concept analyzed is the retrospective, which is often underestimated in Scrum but, in my experience, is the most crucial meeting for fostering a cohesive team working towards a common goal. The Scrum methodology mandates a significant number of hours spent in meetings with the entire team involved, which is why the retrospective allows us to address habits and attitudes that could harm the team.

In the case study provided, the retrospective has also led to tangible improvements, few are described below:

- A meeting called "Code Review" was added every Friday to review and consolidate the work done during the week. This helped avoid accumulating problems related to new implementations during the sprint.

- New labels were introduced for epics to identify those requiring effort during the deployment phase, thus preventing issues during this critical stage.

- A new template for defect creation was shared to the client to save time by providing necessary data for analysis.

- Guidelines for all checks to be performed after the deployment phase were established, ensuring clarity on the process even for new team members.

6.2 Results and Future Steps

The results obtained from the paper can be categorized into two main areas. The first category focuses on the progressive improvement in terms of story points delivered by the team, and consequently, on the revenues generated for the company Reply. The second category concerns enhancements made to the Scrum methodology, achieved through the analysis and study of scientific papers that highlighted certain shortcomings of the method, followed by the implementation of solutions aimed at addressing these deficiencies.

Regarding the first point, a consistent upward trend was observed, resulting in an increase in story points delivered. Initially, the team averaged 6 story points delivered per sprint per member, which later stabilized at a consistent result of 11 story points delivered. While various factors influence a team's velocity, it is evident that the implementation of improvements to the Scrum process had a tangible effect on the project. This is supported by the project KPIs explained in Chapter 5, which also demonstrate collective improvement. From the data collected on the KPIs (velocity, focus factor, percentage of found work, and percentage of adopted work), the trend is positive, with setbacks that highlight moments attributable to team member changes.

The aspect that requires further attention is the quality of the released product. Assuming that the team's code quality has always been of a good standard since the beginning of the project, the increased workload has also led to an increase in defects found in end-to-end testing, resulting in higher rework costs. Processes can certainly be introduced to ensure a higher quality of the delivered product.

As I will continue in the role of Scrum Master working on the project, it will be my responsibility to continue studying and collaborating with individuals with greater seniority to define new project processes with the right flexibility and adaptive spirit typical of the Agile philosophy.

Furthermore, I hope that the document will be useful to future Scrum Masters within Reply, and that they, in turn, will create a similar document for their project, thus forming a database with various types of Scrum methodologies adapted to different sectors.

Bibliography

[1] *The 5 Phases of Waterfall Project Management*

URL: https://www.profit.co/blog/task-management/the-5-phases (cit. on pp. 6)

[2] *Critical Path Method for Project Management*

URL: https://www.wrike.com/blog/critical-path-is-easy-as (cit. on pp. 7)

[3] *Program Evaluation and Review Technique (PERT) Analysis*

URL: https://acqnotes.com/acqnote/tasks/pert-analysis (cit. on pp. 8)

[4] *History of Agile Methodology: How it was Developed*

URL: https://www.knowledgehut.com/blog/agile/history-of-agile (cit. on pp. 10)

[5] *SAFe 6.0*

URL: https://scaledagileframework.com/ (cit. on pp. 10)

[6] *Core Values in SAFe*

URL: https://scaledagileframework.com/safe-core-values/ (cit. on pp. 12)

[7] *SAFe Value Streams*

URL: https://scaledagileframework.com/development-value-streams/ (cit. on pp. 13)

[8] *Agile Release Train*

URL: https://scaledagileframework.com/agile-release-train/ (cit. on pp. 15, 16, 18)

[9] *The Scrum Framework Poster*

URL: https://www.scrum.org/resources/scrum-framework-poster (cit. on pp. 20)

[10] *What is Scrum Team? - Scrum Guide*

URL: https://www.visual-paradigm.com/scrum/what-is-scrum-team/ (cit. on pp. 21)

[11] *Events and Artifacts of Scrum*

URL:https://www.agile-academy.com/en/foundations/events-and-artifacts-of-scrum/ (cit. on pp. 24)

[12] *What are Scrum Artifacts and how to use them?*

URL: https://www.invensislearning.com/blog/what-are-scrum-artifacts/ (cit. on pp. 24)

[13] *An agile guide to scrum meetings*

URL: https://www.atlassian.com/agile/scrum/ceremonies (cit. on pp. 27)

[14] *Reply Company Profile*

URL: https://www.reply.com/contents/Company_Profile_eng.pdf (cit. on pp. 31)

[15] *On using planning poker for estimating user stories,*

Viljan Mahnic

URL: https://www.sciencedirect.com/science/article (cit. on pp.50)

[16] *Spillover items in Agile*

URL: https://www.linkedin.com/pulse/spillover-items-agile-venkat-r-eweic/ (cit. on pp.50)

https://www.learnow.live/blog/what-are-the-4-types-of-operational-value-streams-in-safe

[17] *KPI-Based Approach for Business Process Improvement,*

Aicha Wannesa and Sonia Ayachi Ghannouchia

URL: https://www.sciencedirect.com/science/article (cit. on pp.58)

[18] *Scrum Metrics for Hyperproductive Teams: How They Fly like Fighter Aircraft, Scott Downey and Jeff Sutherland* (cit. on pp.58)

URL: https://ieeexplore-ieee-org.ezproxy.biblio.polito.it

[19] *Risk Based Scrum Method: A Conceptual Framework,*

Nitin Uikey Ugrasen Suman (cit. on pp. 62)

URL:https://www.researchgate.net/publication_Risk_Based_Scrum_Method_A_Conceptual_Framework

[20] *Integrating Risk Management in Scrum Framework,*

Muhammad Hammad, Irum Inayat

URL: https://ieeexplore.ieee.org/document/8616984 (cit. on pp. 62)

[21] *Retrospective*

URL: https://airfocus.com/glossary/what-is-a-retrospective/ (cit. on pp. 66)

List of Figures

Ringraziamenti

Innanzitutto, voglio ringraziare i miei genitori, che mi son stati vicini fin dal primo giorno di questo tortuoso viaggio, e colgo l'occasione per scusarmi per tutti i mal di pancia che li ho fatto passare, ma alla fine, ce l'abbiamo fatta!

Successivamente, ringrazio sia le mie sorelle, Valentina e Marta, che la mia fidanzata, Laura, per avermi sostenuto ed aver creduto in me, e aver così fatto, che un poco, ci credessi anche io.

Ps: grazie Laura non solo per il supporto, ma anche per avermi sopportato in questi anni.

Inoltre, ringrazio i miei amici e i miei compagni di corso della Magistrale, è stata una bella avventura!

Infine, un grazie speciale va a due persone per cui nutro una profonda stima.

La prima è Giorgio, il mio capo in Reply, per aver creduto in me ed aver riconosciuto in me un "agitato" proprio come lui.

La seconda, il mio relatore Marco Cantamessa, per avermi dato sempre gli input corretti per continuare la stesura dell'elaborato e avermi ispirato con il suo corso.