Politecnico di Torino

Department of Management and Production Engineering

Master of Science in Management Engineering

Analysis of the effectiveness of the Scrum approach in the management of an IT project

Supervisor:
Prof. Guido Perboli

Student:
Viola Nuti
286810

Academic Year 2022/2023

Abstract

Scrum is an agile framework for project management which is mainly used in software development, although it is spreading to other fields. It is an iterative approach that ensures that the team delivers increments of the product within fixed time intervals, i.e., *sprints*, which typically last from one to four weeks. Scrum approach deals with the implementation of agile mindset within complex projects, promoting transparency, inspection, and adaptability throughout the process. This study aims to analyze the effectiveness of the Scrum approach in an IT project management, identifying the benefits and the difficulties encountered during the development of the project. The examined case study concerns the integration of spare parts trade with the D2C e-commerce of finished goods, and it is carried out in a worldwide company in the field of domestic appliances. The thesis starts with a smattering of the organization, which is not specified due to strategic and confidentiality reasons. Afterwards, an overview about project management techniques, as traditional and agile, is described basing on literature research. Several agile methodologies are depicted, with the focus mainly on Scrum. Then, a detailed description of the project is offered with the aim of fully understanding the work organization and the functionality of Jira Software, used to handle it. The thesis focuses on the application of the Scrum approach to the project, thus how the framework has been adopted in the managerial process and spotlights any differences. The work elaborates the team performances in the project and proposes an evaluation of the Scrum effectiveness through an interview to the team.

Contents

List of Figures

List of Tables

Introduction

Nowadays, even more importance is attributed to project management discipline since its relationship with project success is widely established. With the purpose of keeping their projects competitive and boosting their profit margins, companies are currently being forced by market competition to act quickly in response to these changes and make organizational and processes adjustments, particularly when the projects are a part of dynamic, uncertain business environments. Furthermore, companies are dealing with more and more projects characterized by a high degree of innovation and volatility, which require strong flexibility and adaptability, also in the way the project is handled. It is in this context that the agile approach to project management has increasingly taken hold in recent decades. Unlike the traditional project management method, which involves sequential and plan-driven phases, agile is an iterative approach that bases its principles mainly on lightness, adaptability, flexibility, and simplicity. Although a lot of studies in literature are struggling to find a decision model to select the best type of project management, only guidelines and suggestions emerged, which mainly rely on the characteristics of the project.

In agile, the product is defined while the project is already ongoing, through incremental releases of value that allow stakeholders to release continuous feedbacks and quickly adapt the product to new requirements. Agile was born as a software development methodology composed of iterative cycles, but its predisposition to cope with changes has meant that it spread in other areas, such as project management. Agile project management, which provides strategies to adjust the process to absorb application, scope, and product feature changes, has in fact recently gained popularity among businesses.

Among the different techniques used to implement agile paradigm, one of the most widespread is Scrum, which is the approach examined in this thesis. Scrum is a framework that enables the adoption of the agile mindset in project management. According with the Scrum Guide, the framework is characterized by

specific events, artifacts and values. It also provides a Scrum team. Scrum ensures the delivery of increments in fixed time intervals called *sprint*, promoting transparency, inspection, and adaptability throughout the process.
This thesis aims to study the effectiveness of the Scrum approach applied to a specific IT project, in attempt to better understand this practice and to offer useful insight to the companies that deal with it. Thus, the work investigates on the main advantages of the Scrum, also highlighting any point of attention. The used research method was a case study in a domestic appliance company, which applied the Scrum framework to develop the spare parts section in the D2C e-commerce of one of its brands. Due to strategic and confidentiality reasons, the name of the company is not cited.
In addition to this brief introduction, the work is organized in six chapters. It starts with an overview of the context of the project, providing information about the company and interesting insights about the D2C e-commerce in the domestic appliance market. Then, a literature research of project management is described in Chapter 2. This theoretical contribution is fundamental to better understand the concept of agile, as well as the differences between traditional and agile methodologies. The most used techniques to implement agile spring also up, with the focus especially on Scrum framework. Subsequently, a deep description of the project is offered in Chapter 3, with main evidence on organizational aspects thorugh the utilization of Jira Software. Chapter 4 deals on how Scrum is applied within the project, analyzing the best practices adopted by the team. Then the performance of the team, as well as the Scrum effectiveness, are examined in Chapter 5. To better investigate on how the Scrum approach is perceived by the team, thus its relationship between project success, a questionnaire has been filled by the team. At the end, final considerations, limitations of the study and potential future research are outlined.

Chapter 1

The Company Context

1.1 The Company in the World

This thesis was carried out in an American company, worldwide leader in manufacturing small and major domestic appliances, with an annual revenue of around $22 billion and around 69,000 employees in 2021.
The company counts four headquarters, each located in a different continent, and around 18 brands that cater to various global lifestyles. Its products are sold all over the world, both in store and online. It includes categories such as laundry, refrigeration, cooking and dishwashing. For some years the company has also been selling finished goods' spare parts, that is the category on which this thesis focuses. Spare parts are primarily sold to parts distributors and retailers, with a small number of sales to end consumers. Together with warranties, spare parts net sales of the company amounted around to $1.2 billion in 2021.
The company is organized in four segments, each referred to a market and managed by a board of directors. In particular, there are North America, Latin America, EMEA and Asia groups.

1.1.1 EMEA Segment

The EMEA group, which includes Europe, Middle East and Africa, is the second largest segment of the company, after the North America one. EMEA segment provided approximately 23% of the total revenues of the company in 2021.
EMEA group counts over 19,000 employees and has 13 industrial and technological research facilities across 6 countries.
The organization operates in this region throughout several famous brands, with

a sales presence in 35 markets. In particular, the company markets and distributes its major and small home appliance to retailers, distributors and directly to consumers within 8 brands in Europe.

1.2 D2C E-commerce

Nowadays the e-commerce is an indispensable part of the global retail framework for most of the industries. Following the development of the internet, the retail environment saw significant transformation like that of many other businesses. As result of the ongoing digitalization of modern life, consumers from almost every nation today benefit from online purchases. Thus, even companies have had to adapt to digital change, such that nowadays the sale of products through e-commerce sites is increasingly widespread.
Looking at the household appliances market in Europe, data show that most of revenue share comes from distributors and other offline sales channels, rather than online ones. Despite this, the trend for online sales seems speedily increasing. As highlighted in Fig. 1.1, online channels may be expected to double their shares in less than ten years.

A very popular business model for online sales channels, and especially for the

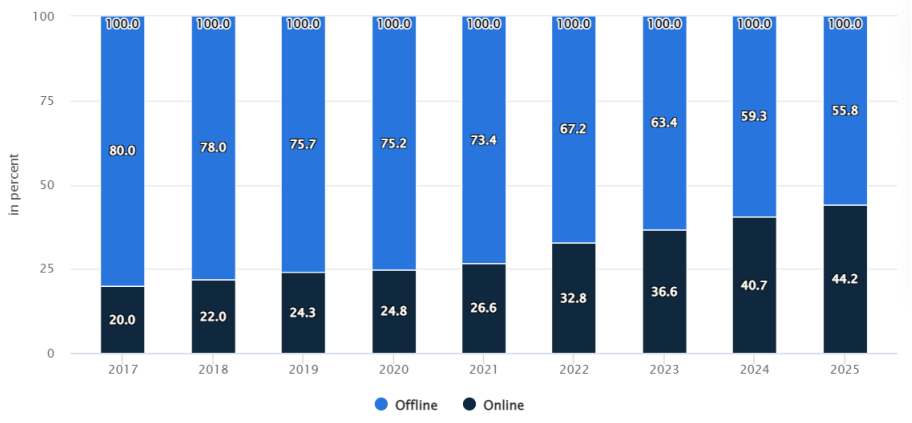


Figure 1.1: Percentage of sales through offline and online channels in Household Appliances Market in Europe (Statista, 2021).

e-commerce, is the Direct to Consumer (D2C). It consists of producing and selling the products directly to final users within own channels, thus without any middlemen. There are multiple advantages in adopting the D2C model. First, avoiding distributors going in-between the brand and its final customer brings the company closer to the audience. This leads the company to get higher control over the marketing and thus over profit margins. Selling throughout D2C

Elements	Quantity
Spare Parts	60,000
Sellable Spare Parts	38,000
Referring Appliances	32,000
Technical Drawings	26,000
BOM Relationships	3,000,000

Table 1.1: Catalogue in numbers.

means indeed to reduce complexity in the sale process and to reach final users fast. Furthermore, D2C sales allow the company to gather an enormous amount of targeted data, with the possibility to take advantage from them. Hence, collecting more information about the customers permits the firm to better adapt the product range, offering a higher degree of personalization and thus increasing profits. Still, the D2C e-commerce is perfectly integrable with other sales channels, both online and physical, gaining higher market share.

1.2.1 Business Case

Consistent with market trends, also the company object of this thesis included in its digital evolution the development of new channels, in particular starting to build the D2C e-commerce platforms for every brand. In this way, customers can buy products directly from the brand website with official warranty and support. The D2C website involves mainly the sale of domestic appliances. However, taking into consideration the importance of the e-commerce, the company decided to start a campaign to also include the spare parts in its D2C channels. The project analyzed in this thesis started here. In particular, it refers to the German D2C e-commerce of a brand operating in Europe. The purpose was to integrate the spare parts into the e-commerce of finished goods that was already active. In fact, the company aimed to reach a larger market share, allowing end-users to buy spare parts for their appliances directly on the website without going to a distributor. In terms of numbers, the integration involved the loading of the catalog, which was of the order of 60,000 spare parts. In turn, they triggered the uploading of around 26,000 technical drawing images that show each component of the appliances. Then, around 3,000,000 relationships needed to be charged to link appliances, technical drawings, and spare parts on the website. The spare parts were related to approximately 32,000 finished goods. After the interaction with the catalog of the prices, only 38,000 spare parts over 60,000 resulted sellable,

because those that were not available and the obsolete without any substitutes were excluded from the website. To sum up, the numbers are collected in the Table 1.1. However, the catalog is dynamic and it is continuously updated.

Chapter 2

Project Management: an Overview

2.1 Introduction to Project Management

The Project Management Institute (PMI) defines a project as *a temporary effort to create value through unique products, services, and processes*. It appears that a project is limited in time, i.e.,*temporary*, with a starting and an end date. The team is also created at the beginning of the project and dissolved at the end. The outcome of the project is *unique*, meaning that it cannot be serializable, with the deliverable that is, thus, defined and definable. However, the project may be stopped before the realization of the outcome, for example for budget issues. Lastly, the project is *progressive*, implying that phases are sequentially constrained among them.
To successfully achieve the goal, the project should be organized and carefully handled in every phase, people involved need to be coordinated and the *modus operandi* must be clearly set. The Project Management discipline includes all the activities related with its accomplishment. Several definitions of Project Management can be found in literature. The one which follows is taken from the PMBOK (*Project Management Body of Knowledge*) [1] and states that:

Project management is the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements.

The Project Management is also defined as [2]:

The planning, organization, monitoring, and control of all aspects of a project and the motivation of all involved to achieve the project objectives safely and

within agreed time, cost, and performance criteria. The project manager is the single point of responsibility for achieving this.

Time, cost, and scope are the three constraints that underlie project management. They also represent the criteria through which the effectiveness of a project is usually assessed [3]. The time relates to the period within which the project must be finished, the cost is the budget that must cover all the expenses and the scope refers to the set of requirements to match, therefore, the perimeter of all actions that must be taken. Those constraints constitute the vertex of the so-called *Iron Triangle* [4], as shown in Fig. 2.1, even if this representation has been evolved over years. A project is then said successful if it has been complete

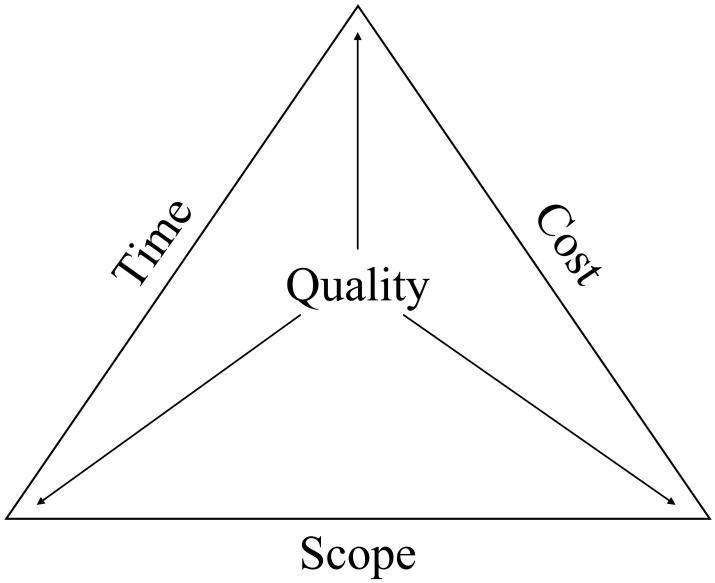


Figure 2.1: The *Iron Triangle*

on time, on budget and if all the requirements match with those expected by the stakeholders. The Project Manager (PM) is the person who has the responsibility that the project respects all the above-mentioned criteria. However, even those constraints are correctly satisfied, it does not directly ensure that the outcome is something of quality. Indeed, quality is also directly involved in a project, and it can be considered as an additional performance criterion. Quality can be evaluated by the workers and stakeholder's satisfaction. Time, cost and scope are strongly related with quality, since none of them can be improved without affecting it. For instance, quality will be lower if the budget is cut without adapting the scope and timing. Further, if the same quality is expected within a shorter period of time, the budget must be increased or the scope cut. The Project Manager's

goal is to find the right trade-off among those elements and quality, in order to ensure a successful outcome to stakeholders. In addition to the Iron Triangle, some studies are raising the possibility that other performance criteria should be taken into account to assess the goodness of the project, as the benefits for the organization and stakeholders [2].
It is widely consolidated that two main approaches of project management, the *traditional* and the *agile*, lead all the other methodologies. However, in real contexts, it is ever more common the application of a hybrid approach, even if literature related to this is still quite scarce.

2.2 Traditional Project Management

Traditional project management is a linear and predictable project planning system where activities must be taken following a logical sequence. Since every phase can only be started after the previous one has been completed, it is also called *waterfall* project management.
In the traditional approach the expected outcome of the project is clearly determined by the client at the beginning of it. It is a plan-driven method, where each phase is carefully defined in the early stage of the project in terms of goals, work packages, responsibilities, and deadlines [5]. Referring to the Iron Triangle, the scope is fixed, established, and well detailed at the beginning of the project through an achievable set of objectives, while the time and budget are defined accordingly [6]. The stability and the formal procedural rigor caused by such a detailed scheduling, represents the strength of the waterfall approach. They also reduce uncertainty and volatility through the entire project since the roles and the process are fixed and well defined in terms of accountability. The systematic and documented planning gives a solid structure to the workflow, and it also allows to monitor the project progress via *milestones*, supporting its control in terms of time and budget. Milestones are tools often used to designate special locations along the project timeline. However, if on one hand the traditional project management emphasizes the importance of planning and controlling, on the other hand it is limited to projects that follow a sequential flow [7]. Indeed, while the timely and long-term planning of the project makes it possible to have a strict roadmap, traditional method has also some disadvantages. Defining in detail the scope and documentation at the beginning could be very challenging, especially if stakeholders give abstract requirements that, in turn, causes incorrect

planning. This is a very critical point, since making changes after the project is started could be cumbersome. Additionally, there is not continuous involvement of stakeholders in a waterfall project, and it does not allow to receive feedback until the project ends. Hence, different peculiarities of traditional approach lead to consider it as successful or not. Of course, it also depends on the kind of the project to be handled. The nature of projects is widely different and not all of them suit the waterfall approach. Indeed, the lack of flexibility and the rigidity of project phases cause the difficulty to adapt the project to new requirements that may arise later. According to this, a more traditional project management should be applied in projects where the scope requirements are clearly provided, fixed, complete and with a low dynamics and uncertainty rate. Hence, the level of innovation of those projects is usually low. Again, traditional project management better suits project that are *one-shot game*, i.e., where frequent variations are not possible from a legal/technical perspective or are linked with unacceptable costs. In terms of project characteristics, it usually fits better large size projects, labelled by large teams, long duration, and complex end product, for example composed by a high degree of interdependencies between the components [5].

2.3 Agile Project Management

2.3.1 Agile Values

The term *agile* appeared the first time in 2001 in software development field, in order to streamline the process from the excessive planning and documentation that were slowing down the entire cycle. During years, it has been claimed to be an iterative project management approach that is currently spreading into several industries. However, the concept of agile is much wider than its application in project management. Agile is indeed a mindset, a philosophy, an organizational paradigm that is based on four concise values that are stated in the Manifesto for Agile Software Development [8]. They are reported below.

- ***Individual and interactions over processes and tools.*** Even if processes and tools are still essential, they are not the only way by which individuals of the team are led in the project. In agile, a more informal and lean communication is expected, empowering individuals and their interactions, rather than rigid and scheduled processes.
- ***Working software over comprehensive documentation.*** In agile

view, setting the delivery of a working increment of the software (or product, or service) is a priority over losing time to document its progress.

- ***Customer collaboration over contract negotiation.*** Hence, investing time to create a solid partnership with a client is more important than spending time in negotiating contracts with detailed constraints and clauses.

- ***Responding to change over following a plan.*** Agile mindset focuses on the ability to be flexible and to adapt the project on changes, rather than strictly planning and scheduling. Planning is important also in agile project management, but it is a short-term planning.

In addition, there are twelve principles that support agile values [8]:

1. ***The highest priority is to satisfy the customer through early and continuous delivery of valuable software.*** Customer's centricity is the starting point of agile. Their opinion is monitored through constant feedbacks, and continuous enhancements to the product must be released.

2. ***Changing requirements are welcome, even late in development. Agile processes harness change for the customer's competitive advantage.*** Flexibility and adaptability to changes are new challenges for achieving the competitive advantage.

3. ***Delivering working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*** Workload is divided into iterations (*time boxes*) with short time horizon that bring value to the project, minimizing effort.

4. ***Business people and developers must work together daily throughout the project.*** Collaboration and participation are the milestones of agile. Project is not simply committed to a team by stakeholders, but they are all involved in its implementation.

5. ***Building projects around motivated individuals, giving them the environment and support they need, trusting them to get the job done.*** Teams have their accountability in the project. It is important to ensure them a safe environment in which to take decisions, to deal with problems and to place trust in other members. In this way, also the proactivity is stimulated, allowing the diffusion of new ideas and changes.

6. ***The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*** According with the importance of informal interactions, the one suggested is face-to-face. Indeed, to focus directly on the topic, without wasting time in waiting for answers, is part of the lean organization, to which agile approach aims.

7. ***Working software is the primary measure of progress.*** To evaluate the performance of the progress, the operational outcome must be taken mainly into consideration. Again, spending too much time on documentations may be a huge waste.

8. ***Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*** In this way, innovation is allowed through continuous interactions and risks are lowered.

9. ***Continuous attention to technical excellence and good design enhances agility.*** Since there is not a final end in an agile project, the technical enhancements basically are what stakeholders are looking for.

10. ***Simplicity, i.e., the art of maximizing the amount of work not done, is essential.*** Being agile means to avoid as much waste as possible in terms of money, time, effort and resources. This principle is also the fundamental of *lean philosophy*.

11. ***The best architectures, requirements, and designs emerge from self-organizing teams.*** Autonomous and independent teams are the best way to organize the work, since they are enabled to make fast decisions and to better fit the change.

12. ***At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*** In order to achieve continuous improvements, team members are used to give feedbacks in specific moments, thus learning from their errors.

According with values and principles, agile project management is characterized by a scope that is not fully agreed in the early stage, rather it is defined step by step by stakeholders while the project is still ongoing. In this case, the stable vertices of the Iron Triangle (Fig. 2.1.) are time and budget, while the requirements

are dynamic [6]. The peculiarity of the agile approach is that value is bring to the outcome in an incremental way, releasing small and constant improvements to the product. To do this, the project must be apt to be divided into distinct iterations of work, as in software development [9]. This approach allows to add, delete, and frequently modify the features of the project or simply to postpone their definition. Thus, the ability to fast and easily adapt the project to changes is the crucial point of agile approach, while the rigid adherence to the initial plan, fundamental on waterfall, is here secondary [10]. This mechanism is enabled by the fact that stakeholders are strictly involved in the project, and they can constantly update the requirements through continuous feedbacks. It also eases the process optimization and its continuous improvements, lowering the risk of false development [5]. Since the development cycles are short, the identification of errors is usually quick. Another peculiarity of agile approach is the project team. It is composed by people who are self-organized and thus the level of collaboration, coordination, learning and adaptation among team members must be high. Communication between stakeholders and team members is also frequent and usually informal. Furthermore, people with higher technical skills related to the project scope are required. Due to their autonomy in the project, they perceive a greater involvement and responsibility and, therefore, their motivation on average is high [5]. On the other hand, a strong dependency among project success and team self-organization is triggered and this can be risky if team members are not used to experience agile.

Although agile adoption is growing in several industries, it also has some weaknesses. First, the lack of pre-established requirements may cause difficulties in estimate the exact time and budget. If enough effort is not put in tracking the process, it is also difficult measuring the progress, since it dynamically changes through cycles and iterations. In addition, a good communication is crucial in any agile project, but it is not always efficient, especially in the case of large and distributed teams, causing lack of cohesion in deliveries. Thus, depending on the type of the project to be handled, the agile approach can be more or less appropriate. How to decide if to apply waterfall or agile project management is widely discussed in literature but strict rules have not yet been come up. In general, an agile apporach better fits projects which have a large predisposition to be decomposed, so those whose solution can be carried out in increments. Furthermore, projecs that require frequent adjustments and modifications during the development phase are better suited to an agile mindset. As already explained,

the flexibility is basically needed to compensate for the impossibility of collecting formalized requirements at the beginning of the project. According with literature, agile project management is commonly used in medium-small project with a low degree of complexity and a certain level of novelty and uncertainty, since in this way it is easier to respond to changes [5]. The choice if implementing agile should be also related with the characteristics of the organization itself. Indeed, it must be aligned with the agile mindset, embracing fully the philosophy and the principles behind. Also, it must be able to regularly deliver increments.

To sum up, traditional and agile are two complementary manners to manage a project. In general, preferring an approach rather than the other depends on the intrinsic characteristics of the project that must be dealt, as previously explained. However, actual projects are even more complex than theoretical ones. A project may have features that would better fit with an agile approach as others that are perfect for traditional method. As a matter of facts, it is even more common that organizations practically follow a hybrid approach, with peculiarities coming from both waterfall and agile. Literature related with this topic is still scarce, but, according with [9], a significative percentage of projects seems to adapt lean characteristics with a rigorous planning. It is interesting to notice that the outcome is mostly successful as well.

2.3.2 Agile Methodologies

In literature, several methodologies exist to implement agile mindset. Among all, the most common are Extreme Programming (XP), Kanban, Lean, SAFe and Scrum. Since this thesis focuses on Scrum, it is deeply explained in Sec. 2.4. For completeness, a brief overview of the other methodologies is also provided below.

Extreme Programming (XP)

Extreme Programming (XP) is a framework agile for software development proposed by Kent Beck. It stands on a set of principles, values and practices that aim to create a good-quality environment for developing software that allows the generation of simple codes that are, in turn, easily adaptable to changes.
As an agile approach, XP provides iterative and incremental cycles that usually last one or two weeks, called weekly cycle, during which the team and the client agree on which task to implement. Then, they release a demo at the end of the cycle to collect feedback from stakeholders.

Differently from Scrum, members of the team are not clearly accountable for some tasks but rather XP strongly believes that everyone should contribute at the best effort, independently by their specific functions. On the other hand, XP values are very close to the Scrum ones. In particular, XP is characterized by simplicity, communication, continuous feedback, courage and respect.
However, XP may be few efficient if the project involves many people and the lack of a register of errors increases the risk they can be repeated.

Kanban

Kanban is not a software development lifecycle methodology, as XP, or an approach to project management, as Scrum. It is required that a process is already in place, so that Kanban can be applied to incrementally change the underlying process [11] and increase its efficacy with the support of the *lean thinking*. The core practices on which Kanban is based are [11]:

- Visualize Workflow;
- Limit Work-in-Progress (WIP);
- Measure and Manage Flow;
- Make Process Policies Explicit;
- Use Models to Recognize Improvement Opportunities.

Kanban, that literally means *card*, is a visual model that mirrors how the work is going. It strongly states that the visualization is the capstone to better handle a process or a flow. In Kanban, it is preferred to face one thing at a time and finish it, rather than open many tasks at once running the risk of leaving them inconclusive. This is consistent with the goal of releasing small but frequent amount of value to the clients. The importance of measuring and managing the flow involves the concept that the value is incrementally created in every single instant, and it must be carefully monitored with the purpose of avoiding bottlenecks.
Kanban aims to support the team in deeply understanding the flow of the work, so that it may be able to manage it more profitably. The instrument of excellence used in this methodology is the Kanban Board, i.e., a table where, depending on their status, activities are usually moved under the columns *to do*, *in progress* or *done*. Although the standard Board includes the previous columns, it can be also

transformed and customized, depending on the needs. The Board allows a clear visualization of the project trend. Generally, it includes the Swim Lane, that is a space for reporting activities that should be accomplished urgently. The WIP indicates the threshold of the maximum number of activities that simultaneously can be in a specific status. Hence, the team is obliged to complete activities that are already in the board, before introducing additional work. The choice of the WIP is important, given that a WIP too high may risk a gridlock in the process. In conclusion, Kanban is incredibly adaptable because it does not call for distorted team roles or processes. On a more basic level, it seeks to raise awareness of the workflow and encourages the team to continuously work on its enhancements through feedbacks.

Lean Development

More than an agile methodology, the Lean Development is a proper mindset that includes a set of values. The key for this philosophy is the elimination of all that represents a waste and does not add value to the final product, both because it is not necessary, and because it is inconsistent in workflow or excessively difficult. Lean Development gives value to the team within the whole process, in such a way each member is aware of the project from a high-level perspective. Strictly related with Lean, there are concepts like *pull system* and *just-in-time*. They relate with the fact that production should not be led by forecast but rather by market demand. Thus, the product should be releases as fast as possible to collect quickly feedbacks and make enhancements.
As already mentioned, Lean Development is a mentality, a set of values, but does not provide practical techniques that the team can adopt. It offers the theoretical guidelines to be followed to implement an agile approach.

SAFe

The Scaled Agile Framework (SAFe) consists of a set of organizational and workflow patterns used to scale agile at an enterprise level. The goal is to encourage coordination and delivery across many agile teams and achieve the business agility, such that the company can properly face with market volatilities and demand changes.
The first version of SAFe has been released in 2011 by Dean Leffingwell and Drew Jemilo, but it is still evolving. Among all SAFe's core values, those more meaningful are alignment, built-in quality, transparency, program execution and

leadership. The more recent version is SAFe 5, that provides seven core competencies to understand and implement SAFe. They are described below [12].

- ***Team and Technical Agility***. The agile team represents the starting point to introduce agile to larger enterprise. Teams are cross-functional, self-managed and are autonomously able to develop solution within a time. The team uses methodologies as Scrum, Kanban or hybrid. To scale at enterprise level, more teams work together and coordinately. In SAFe, the team of agile teams, responsible for delivering solution outcomes, is called Agile Release Train (ART). ART works on specific periods called Program Increment (PI), that are, in turn, divided into five iterations.

- ***Agile Product Delivery***. In this approach the clients are placed at the center and the goal is to offer them the best solution at the right moment. Although the solutions are constantly developed, their release occurs only when the client's need arise. To be sure the characteristics of the product match the client's requirements, several techniques are used, as instance *design thinking*.

- ***Enterprise Solution Delivery.*** Its function is to align and coordinate more ARTs, in case of large organizations.

- ***Lean Portfolio Management.*** This competency aims to modernize Portfolio Management within a Lean-Agile approach, aligning strategy and execution.

- ***Organizational Agility.*** It is the enterprise's capacity to self-reorganize quickly in front of market changes.

- ***Continuous Learning Culture.*** SAFe provides a set of values and practices that encourage continuous enhancements.

- ***Lean-Agile Leadership.*** In order to uniform all the enterprise with agile, it is essential that the leadership embraces, spreads and teaches lean mindset within the entire organization.

2.4 Scrum

2.4.1 Scrum Values

Among all the different approaches that can be followed to set an agile mindset in project management, one of the most widespread is Scrum.
Scrum was shown for the first time in 1995 by Ken Schwaber and Jeff Sutherland and six years later the *Agile Software Development with Scrum* book was published. The term *scrum* comes from the rugby and, as in this sport, people involved in a project run together, quickly passing the ball, pushing towards the common meta. Even for agile, it was born for software development, but it is current applied in several field.
It is described as a development process for small teams [13], which includes a series of short development phases, called *sprints*, that can last from one to four weeks. The team captures identified tasks in a backlog which is reprioritized and updated at the beginning of each sprint [14].
More precisely, the Scrum Guide [15] provides the following definition of Scrum:

Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.

It is defined as a framework, not as a methodology, since the latter is more detailed and structured than the former. Scrum is, indeed, a set of artifacts, events, practices, and accountabilities that are purposefully incomplete, leaving a lot of room for intervention. Therefore, it can be adapted by experience. Scrum focuses on simplicity, lightness, and flexibility. It is also strictly related with the *lean philosophy*, since the priority is to minimize waste, concentrating only on the essentials. It allows teams to self-manage, giving them more independency and freedom to take decisions faster, adapting to changes without losing time and effort in asking permissions. In this way, downtimes are avoided, and the process can proceed more smoothly. Another source on which it is based is empiricism, since experience and the capability to make decisions based on what is observed are the key elements to be lean. Furthermore, learning from experience and enhancing by doing push towards continuous improvement, that is another pivotal element of agile thinking.
The Scrum Guide provides three pillars on which the Scrum empiricism is based. They are:

- ***Transparency***: process and artifacts must be clear and visible both to those who work there and to those who benefit from it. Low transparency can mislead to wrong decisions that, in turn, may decrease value and arise risks. Transparency enables inspection.
- ***Inspection***: artifacts and the progress must be monitored and inspected frequently and diligently to detect potential changes or issues. To do this, Scrum provides five events that should be taken regularly, as it will be explained in Sec. 2.4.4. In turn, inspection enables adaptation.
- ***Adaptation***: if variations are coming from the inspection, process must be adjusted as soon as possible in order to avoid further deviations. It is permitted by self-management of the teams.

In addition to the above-mentioned pillars, the Scrum Guide also states five values that mainly lead the Scrum framework. They are:

- *Commitment* to achieve the goal by the team and support each other;
- *Focus* on the task at hand and make every effort to move the project forward;
- *Openness* about the work and the challenges;
- *Respect* for other team members and their capabilities;
- *Courage* to do the right thing and to deal with problems.

2.4.2 Scrum Framework

As already said, Scrum is a light framework that mainly consists of iterative practices. As shown in Fig. 2.2, the flow is quite simple. Starting from the stakeholders' inputs, a list of actions fulfils the Product Backlog. Then, the planning of the activities that must be accomplished in the next sprint are defined and reported in the Sprint Backlog. During the sprint, the goal is to successfully finish all the tasks. The team is constantly updated through daily meetings, and it tries to solve in an autonomous way every issue that may eventually arise. At the end of the sprint, two meetings are scheduled: the Review, in which stakeholders are informed about the *increment* in the value of the product that has been released; and the Retrospective, in which the team members discuss about how the process is going. Subsequently, items in the Product Backlog are

revised, modified, and the new ones inserted. Then, a new sprint starts, and the loop is repeated. Every component of Scrum framework is deeply explained in the next sections.

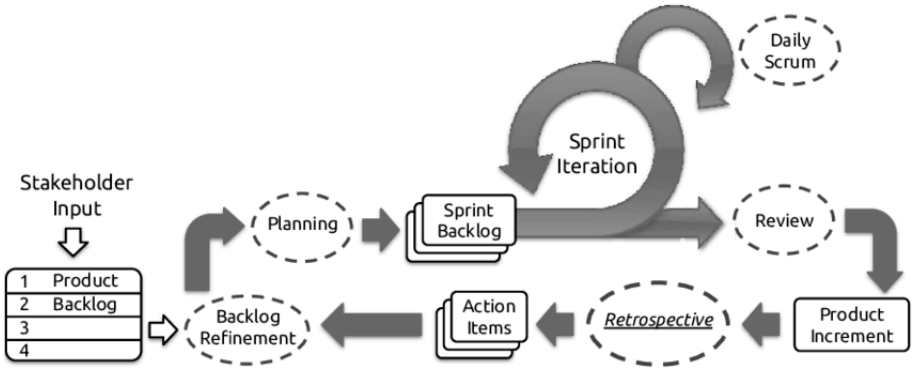


Figure 2.2: The *Scrum Framework* [16]. Scrum Events are represented by circles, while process artifacts and outcomes as squares.

2.4.3 Scrum Artefacts

Artefacts represent the work and the values the team must take charge of. They serve to ensure transparency of information, enabling inspection and, thus, adapt the process through continuous replanning.

Product Backlog

The Product Backlog (PB) is a list of all the actions, requirements and functionalities that needs to be done to satisfy the purpose of the project. Every component in the Product Backlog is called *item* (PBI). They represent small and detailed action into which the requirement can be broken down. Each item brings value to the product. A common method to carefully describe them is through a *user story*. As better explained in Chapter 3, it is a brief sentence that recap the required functionality from a user point of view. They are helpful because allows the team to understand which are high-level functionalities that the products must have from a business perspective, rather than the technical characteristics. They should be short, simply, and clear. However, in the Backlog there might be other elements, such as ideas, bugs, features or epics.
The backlog needs to be dynamic, in line with agile thinking, which emphasizes flexibility and adaptability throughout the project. Items must be modified to better fit environmental changes. When new requirements emerge, they must be added in the PB, and, when further details arise regarding an item, this must be

updated. The team must have always access to the list and all the items should be clearly understood. The items are prioritized and those with higher priority are put on the top of the list. Those are the items on which developers must work first. Backlog is the only kind of documentation Scrum provides, consistently with the agile value that privileges *working software over comprehensive documentation*.
The Product Goal is the long-term objective for the Scrum team, and it is in the Product Backlog.

Sprint Backlog

The Sprint Backlog (SB) represents a subset of Product Backlog items that should be fulfil within a specific sprint. Hence, it is composed by items with higher priority. Basing to items effort estimation, the available capacity (in terms of *man-days*) is filled, and the sprint is planned. The Sprint Backlog is an actionable plan for the developers that allows to release one or more increments of the product, thus increasing its value. It can be considered as a real-time picture of the work that the developers are performing, and it is useful because it allows the monitoring of the project progress. Furthermore, the SB allows each developer knows which tasks must be developed, ensuring them the possibility to self-organize. The Sprint Backlog usually involves a board, either physical or virtual, where items are classified in relation to their status. For instance, an item can be put in *to do*, *in progress* or *done*. In order to successfully close the sprint, all the items should be developed and setted as completed.
In every Sprint Backlog there is a Sprint Goal, which is the purpose of the specific sprint.

Increment

Increment is the value added to the final product in every sprint. Increments represent all the outcomes in a sprint that respect the Definition of Done (DoD). The latter is a formal description of the state of the increment when it meets the quality measures required for the product [15]. DoD enhances transparency among the project, given that it establishes a set of conditions that must be met before marking an increment as done. In this way, the process is unified and standardized, and the entire team needs to be conformed with the DoD. If the Definition of Done is not matched, the item cannot be released, and it returns to Product Backlog for further analysis. Each Increment is usable and valuable, thus

stakeholders can already give feedback on it. It is additive to all prior increments and the sum of all the outcomes in the sprints should be match the Product Goal. To sum up, an Increment is a concrete step toward the Product Goal.

2.4.4 Scrum Events

In Scrum framework, there are four formal *events* where the team members meet and discuss on what they are working and how they are proceeding. During these events, inspection and adaption phases take place, allowed by the transparency that should be present in all of them. Events are all contained in a sprint, which corresponds to one iteration cycle [6]. Within this time interval, an increment in value is expected. The idea behind each sprint it to deliver valuable and usable functionality [13]. In other words, it is the period in which the Sprint Goal should be achieve. Hence, the entire project is divided into iterative time-boxed development [13]. Every sprint must have the same duration and it must be agreed at the start of the project, without the chance to be changed until the end. Depending on the team and the work, it can last from one to four weeks and when it finishes, a new Sprint starts immediately, with the repetition of all the events in an iterative way. Longer duration may cause complexity and increase risk, while shorter sprints can generate more learning cycles, limiting risk, in terms of cost and effort, to a smaller time frame. During the sprint, the Product Backlog is refined, and adjustments are done, basing on further clarifications with the Product Owner.
According with the Scrum Guide [15], in each Sprint there are four Scrum Events: the Sprint Planning, the Daily Meeting, the Sprint Review and the Sprint Retrospective.

Sprint Planning

The first Scrum Event in a sprint is the Sprint Planning. The purpose is to elaborate the requirements and identify the dependences and obstacles, with the aim to successfully plan the sprint. In this moment, the team discusses about the increase in value that the current sprint should bring. Once finalized the Sprint Goal, items are discussed, both in terms of effort and priority, and those that should be included in the next sprint are established. Items in the Product Backlog are usually decomposed in smaller loads of work of one day or less and, for each of them, the developers plan how to create an increment that meets the Definition of Done. Also, the boards and tools are set up. Finally, the Sprint

Backlog is defined. Although the Sprint Planning is a crucial event, the Scrum team can sharpen these items during the sprint. In a nutshell, the decision of what working piece will be produced, that will bring value to the client, occurs the Sprint Planning event [6]. The allocation of time for the Sprint Planning depends on the length of the sprint: for a one-month sprint, it is planned to be eight hours maximum.

Daily Scrum Meeting

The Daily Scrum Meeting is an everyday occasion to inspect and monitor the progress of the sprint, adjusting the Sprint Backlog if problems arouse. It must be short (usually 15 minutes) and done always at the same time and in the same place, in order to make it a routine. It is also called Standup Meeting, as to simulate that it happens standing up and that, therefore, must be a quick daily update. Here, the performance of the work is made transparent by developers, which autonomously decide how to organize their tasks. The Daily promotes communication, synchronization, self-management, concentration and quick decision-making, with the aim reducing the need for other meetings and, as consequence, being more *lean*. It also enhances team building [13] since it helps to create relationships among members.
The Sprint Review is planned to last a maximum of four hours for a one-month Sprint.

Sprint Review

At the end of the Sprint, the outcome delivered and the progress toward the Product Goal is presented to stakeholders and all new information from the sprint just completed are reported. During the Review, opinions and feedbacks are collected and the Product Backlog is eventually adapted to fit new requirements. At this meeting, anything can be modified. According with variable scope expected in the agile approach, work can be added, eliminated, or reprioritized [13]. The Sprint Review shows the sum of the Increments, thus sustaining empiricism.
The Sprint Review is planned to last a maximum of four hours for a one-month Sprint.

Sprint Retrospective

The Retrospective meeting is planned at the end of each sprint. The purpose is engaging team members, in order to increase productivity, quality and, therefore, satisfaction. It is an informal opportunity in which the team analyses themselves and discuss what went well during the last sprint, what problems aroused, and which actions have been taken to solve them. Still, the team identifies the most helpful methods to improve its effectiveness that should be already put in practice in the next sprint. Therefore, also in this event, Scrum values as transparency, inspection and adaptation are essential to reduce risks, enhance working processes and maximize results.
The timeboxed for the Sprint Retrospective is a maximum of three hours for a one-month sprint.

Product Backlog Refinement

The Product Backlog Refinement (PBR) is the process through which items in the Backlog are polished and precisely explained. Every detail is elaborated, too. As a matter of fact, items are transformed from high-level activities into functionalities that can be implemented. During the PBR, new features appear in the PB, both as result of the process of decomposition of raw macro-features, and because of the inclusion of new items resulting from new discoveries.
Even if the Scrum Guide does not consider the Product Backlog Refinement as a Scrum Event, it is an important moment where business and team collaborate, and it can be included as the further implementation of the agile principles [17].

2.4.5 Scrum Team

The Scrum Team is small, typically ten or fewer people. It is a cohesive unit with the same objective (the Product Goal), without sub-teams and hierarchies. The team is cross-functional and self-managing. The first characteristic indicates that, all together, the members have all the skills to accomplish value in each sprint. The latter feature states that the team is structured to be internally managed, thus the Scrum team is allowed to establish who does what, when and how, independetly from the organization. The creation of small and self-paced teams is a milestone in Scrum, since it enhances communication, adhesion, productivity, motivation, and autonomy.
The Scrum team is accountable for creating a valuable and useful Increment every

sprint. It is also responsible for all product-related activities from stakeholder collaboration, verification, operation, experimentation, research, and development. The team must understand stakeholder's requirements and develop them.
The Scrum team is composed by the developers, the Product Owner, and the Scrum Master.

Product Owner

The Product Owner (PO) is accountable for maximizing the value of the product resulting from the work of the Scrum team. PO is also responsible for the Product Backlog, which must be handled carefully. First, PO must clearly transmit the Product Goal, then, define all the items in the Product Backlog, prioritizing and ordering them. The Product Owner must also ensure that the Product Backlog is transparent, visible, and understood [15].
The Product Owner's role is to represent all the needs of the stakeholders in the Backlog and be sure all the requirements will be satisfied. The PO is also the only one that has the authority to cancel the sprint in case the Sprint Goal become obsolete. Furthermore, the Product Owner is the person in charge of accomplishing the PBR, with the help of the team.
It is important to have a PO that is constantly involved in the project, with a clear and full vision of the project and able to be always updated. In this way, the likely that the project may be successfully completed increases. Due to this, delegation of those activities to others, even if it is accepted in Scrum, is not highly recommended.

Developers

Developers are the professionals involved in the creation of the Increment [6].
They work strictly with the PO -and other stakeholders- with the aim to better understand the implementation details [17]. They collaborate following a co-operative approach and the development team must point out to the PO if there are some choices that are sub-optimal from an implementation perspective, that can be further improved. Making decisions about what to implement is not their goal; instead, they should concentrate on figuring out the best strategy for doing so. Indeed, they are in charge of the technical decisions, including what architecture to create and how to implement it.
As reported by the Scrum Guide [15], developers are accountable especially for four points, that are: creating a plan through the Sprint Backlog, increasing

quality adhering to a DoD, adapting their plan continuously toward the Sprint Goal and holding each other accountable as professionals.
They are the main characters in the Daily Scrum, in which they discuss about the progress toward the Sprint Goal and plan activities for the next day of work. It is essential that the development team is self-managed.

Scrum Master

In a nutshell, it is possible to state that the Scrum Master (SM) is responsible for promoting Scrum, enforcing its rules and giving training within the team [6]. The SM is the leader in charge of securing that the Scrum approach takes place as defined in the Scrum Guide, as ensuring that all the Scrum Events occur successfully. Scrum Master is also accountable for the team's effectiveness, supporting the team members and the organization. To do this, the SM promotes cross-functionality and self-management among the team members, helping them to create high-value Increments and meeting the DoD. The Scrum Master also avoids that some impediments may obstacle the progress of the project.
This figure is accountable for keeping the documentation updated and assisting the PO with handling of the Product Backlog, the definition of the Product Goal, and, in general, the management of the team. The Scrum Master is the connecting bridge between stakeholders and Scrum teams and thus, SM must help PO to ensure that every item in the Backlog is clear to developers and consistent with stakeholders' requirements.
The Scrum Master is also essential to aid the organization through leading, coaching, advising, and planning the organization in its Scrum adoption, ensuring that every member deeply understands the philosophy that lies behind the Scrum setting.
Since there are no set regulations that can be simply put into practice to carry out its duties, it is a very multitasking role.

Chapter 3

Case Study: an IT Project

3.1 Description of the Project

3.1.1 Scope of the Project

The project under examination consists of the integration of spare parts catalog in the D2C e-commerce of domestic appliances and accessories for an important brand operating in Europe. The online purchase of spare parts was already possible. Nevertheless, the scope of the project was to build a dedicate section of spare parts in the D2C website, unifying the design with the brand guidelines and increasing the performance. The e-commerce on which it was worked delivers only in Germany since the reference market is the German one.
Despite projects in which new enhancements are continuously released, this can be considered as a real building project, meaning that new pages are developed almost from scratch. Since they had to be released coordinately, a specific date to go live was scheduled when the project started. The go live was set for February 1st, 2023, even though it was originally scheduled for two days before. This day marks the release in production, i.e., the developments are available in the live website. Of course, it is not possible to go live in a limited period with a flawless website and it is not even important for the purposes of the business. In fact, with the aim of adding continuous value and reducing waste, the go live was only planned for an MVP of the project. A version of a product called the Minimum Viable Product has just enough functionality for it to be useable and, thus, to go live. With regard to the project, the purpose was to implement an MVP that allowed the acquisition of spare parts on the D2C website, respecting certain criteria and ensuring a satisfying user experience, both on mobile and desktop.

After the integration has been done, the website handling has been delivered to other two teams. One in charge of developing tasks not considered as primary and deploying continuous enhancements. The other one employed in maintenance, fixing bugs.
This was not the first integration project promoted by the company. The idea behind was indeed to uniform all the D2C e-commerce for every brand operating in Europe. From a strategic perspective, standardizing the best template allows to increase the overall company value, as well as to improve the performance and thus to raise the revenues. Thus, the project for the German brand was essentially a roll-out of the spare parts pages implented for the brand operating in UK. More in details, the objectives of the project can be summarized as those established in the kick-off meeting. The main ones are presented within the following lines.

- Enabling sales of spare parts on the current D2C website of finished goods, through the integration of spare parts catalog.
- Shaping the current UX/UI to have a harmonic user experience, using the brand guidelines to restyle the look and feel for spare parts pages.
- Leveraging Search Engine Optimization (SEO) best practices to improve new pages performance.
- Leveraging Google Analytics (GA) best practices to measure pages metrics.
- Implementing a proper shipping policy and invoices.
- Adapting the purchasing experience into the check-out, as for the shipping and mixed order options.
- Implementing an MVP within the go live date, offering measurable KPIs to drive continuous improvement.

Furthermore, all the requirements must be correct from a legal point of view.

3.1.2 Customer Journey for Purchasing Spare Parts

The project has been designed at high level to optimize the user experience as much as possible, offering an easy customer journey for the purchase of a spare parts Fig. 3.1. The project team was in charge to develop all the pages and functionalities related with the spare parts, as those presented in the customer journey. The new pages released at the end of the project are described in this

section, briefly retracing what is usually the path of the user within the site. Starting from the entry points to enter the spares section, the project saw the implementation of two new ones. The entry points are in the page's *header* and *footer*. The first term indicates the top part of the website layout, whereas the latter the bottom one. Header and footer are the same within all the pages of the website. Pressing one of those buttons, the user is addressed to the *landing page* of spares. This page offers two paths for finding a spare. The first one, which is the one more sponsored by the UX/UI designer, is through the searching by the associated finished good. It consists in filling the search bar with the industrial code (IC) or model number (MN) of the finished good. Every combination of IC and MN identifies a unique finished good. Those codes can be found on the appliance. On the other hand, it is possible to search the spare parts by selecting a product category. As shown in Fig. 3.1, choosing the first path, the user lands on the BOM page. The *bill of material* (BOM) page is a dedicated section for spares with the purpose of speed up their exploration. Here, the customers can use the appliance's line drawings to find the part they are looking for. The customers can also use subcategories to narrow down the search field and select the desired component. Selecting a category in the landing page, instead, leads the user to the *product list page* (PLP). In this section, it is possible to refine the search through the utilization of several filters, as long as the spare parts is found. It is also possible to insert the IC or MN and to reach again the BOM page. Once selected the item, the *product detail page* (PDP) is displayed, nevertheless the path chosen to find it. In this page, the product specifics and, usually, its pictures are shown. Furthermore, the user can check if the part fits the appliance inserting the IC or MN, as proposed on the website. Thus, the user can add it to the cart and finalize the purchase in the checkout section.

The user has the possibility to search for the spare parts directly from the home page, too. In fact, by typing the specific code that identifies the spare, or a keyword, in the native search bar, the system redirects the user to the PDP.

A spare part can have four statuses in the website, basing on its availability in the warehouse. The product is *in stock* when it is immediately deliverable, *out of stock* when it is not currently available, *obsolete* when it is not in production anymore, and, *limited availability* when there is an amount of the product under a certain quantity. The statuses are shown both in PLP and PDP. Users can look for a substitute in case their spares are obsolete since it has been agreed to upload only obsoletes with substitutes.

The functionalities described are the main ones that involved the project. The customer journey reported in 3.1 is semplified since there are minor implementations whose detailed description is far from the scope of this thesis.

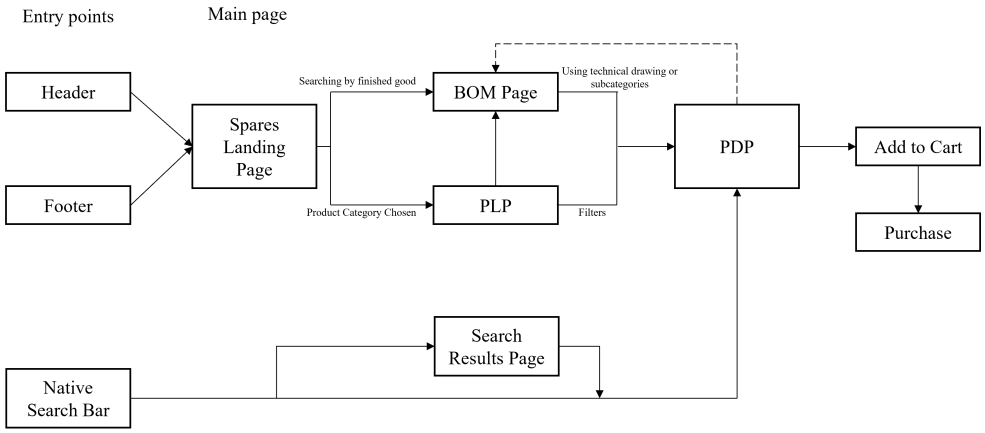


Figure 3.1: The customer journey for the purchase of a spare parts after the integration.

3.1.3 Platforms

The integration of the spare parts in the D2C e-commerce has required the involvement of several platforms. Vtex is the e-commerce provided for the brand and it involves a front-end (FE) and a back-end (BE) side. The considered project aims at both the development of the front-end side, that is the presentation layer, and the back-end, i.e., the data access layer. Generally speaking, the front-end concerns every component that is manipulated by the user, for this reason it is considered as a client-side development. The back-end is instead a server-side code, and usually handles data storage and business logic. Then, the Google Cloud Platform (GCP) is the suite of cloud computing services chosen by the company for the data storage and their managing. It acts as the integration layer between the FE, BE and content layers. ServiceNet is the source of the spare parts catalog.

ServiceNet provides the spares bulk database to GCP. GCP receives products from ServiceNet already pre-filtered, i.e., removing the obsolete product without substitution. It also receives prices, market, and availability conditions from SAP. Afterwards, GCP elaborates all the received information and passes their composition to Vtex. The process is semplified in Fig. 3.2.

Vtex also includes a Content Management System (CMS) that allows to create, manage, and modify web content without using any code.

Salesforce is instead the software used to handle the Customer Relationship Management (CRM). It is the platform closer to the clients, communicating directly with them. As instance, Salesforce allows to send the email to the client after a purchase is made.
Finally, Google Analytics (GA) is the tool used for collecting data from the website and analyzing its performances, as well as tracking statistics for the Search Engine Optimization (SEO) and marketing purposes.
The mentioned platforms are the main ones and those that have been more involved in the considered project. However, there are other platforms that have been required in order to successfully perform the goals.

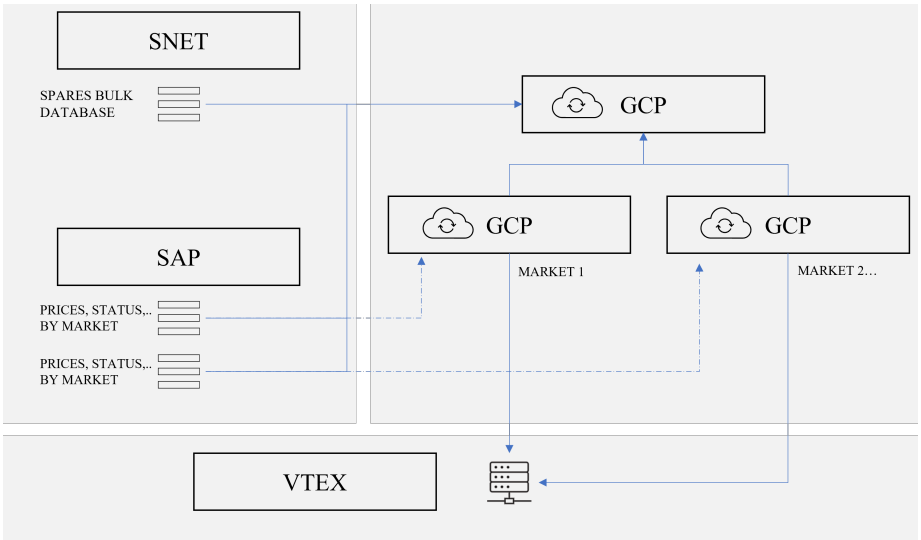


Figure 3.2: The catalog data source for spares parts.

3.1.4 High-Level Planning

The project officially began with the kickoff meeting with the German team on 14th November 2022. The little amount of time available before the go live has immediately been noticed and thus a thorough organization plan was established. While the team was still involved in another project, the collection of the business requirement (BR) started. In order to go live on 1st February 2023, the Gantt chart in Fig. 3.3 has been drawn. The period is divided in six sprints, each lasting for ten working days, except for the last that is longer.
After the BR collection, that lasted more than the time expected, the developments could start. Afterwards, the testing phase had to be done. In particular, the plan expected two weeks for System Integration Test (SIT) and one for User

Acceptance Test (UAT). During System Integration Test the entire order flow is tested, taking into account every combination. Here, a payment was simulated in QA environment. It was also checked if the order is correctly tracked from a back-end and GCP perspective. In addition to technical tests, User Acceptance Test are also important. Users are given the opportunity to interact with the website before it is officially released and thus to check if any features properly works. Everything must be acceptable from the user's standpoint. UATs were led by business local team. Notice that UATs serves to test functionalities already developed, not a moment where new requirements may arise.
In the meanwhile, cutover activities were done to launch the developments in production. Finally, after the go live, the last sprint was dedicated to the *hypercare* period, during which only enhancements were developed, and bugs fixed.

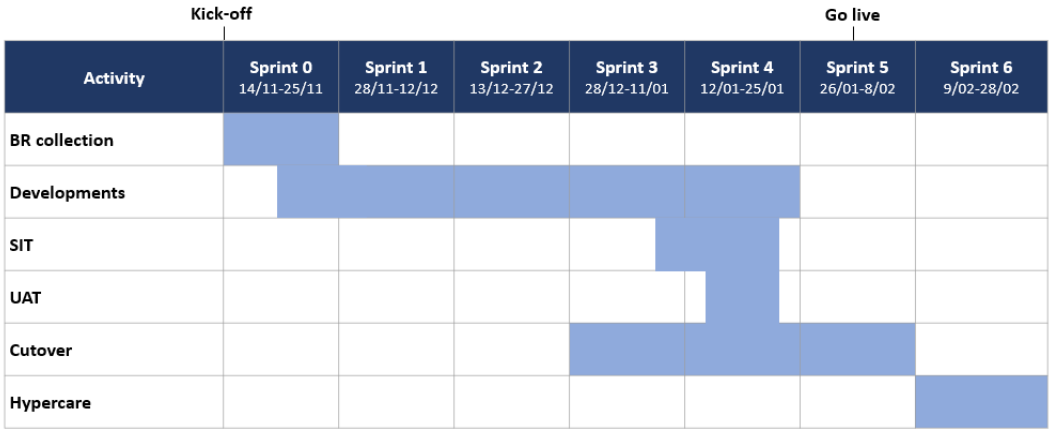


Figure 3.3: The Gantt chart that describes the project phases at a high-level planning.

3.1.5 Core Team

Since it is an IT project, the Global Information System (GIS) department is in charge of it. This department is overseen by the Central district of the Company, not by the local one. A continuous dialogue with the German team was therefore necessary to facilitate comparisons, make decisions and gather feedbacks. The GIS core team accountable for the project is composed by the following members.

- Project Leader or GIS Lead, that is, the person who leads the specific project.
- Product Owner (PO), that is, the person who leads the project from a business perspective. PO represents the business goals, with the aim of

maximizing the value of the product and ensure periodic increments of value.

- Business Owner (BO), that is, the person responsible of the project from a customer service side. As the PO, the BO also collects requirements from the local team.
- Business Analyst (BA), that is, the person who translates the business requirements into technical features.
- Scrum Master (SM), that is, the person who manages the agile project, putting into practices the Scrum principles.
- Developers, that are, the people in charge of developing the required activities, verifying the produced codes are clear and readable. The allocated capacity for the examined project was three developers for the front-end side, two for the back-end side and two for GCP side.
- Tester, that is, the person who must test that every development is properly working.
- Program Management Office, that is the person responsible for reporting the progress of the project to a higher level.

As it can be noticed, it is a numerous team. Due to its importance, the project also needs continuous involvement of other teams, especially the local one. Indeed, its intervention is required every time a decision related to the specific brand must be taken. Specific meetings were set to discuss and to find a solution about fulfilment, assortment, checkout functionalities. The UX/UI designer is continuously asked to provide the most appropriate features for the website. Furthermore, weekly meetings with both the team in charge of improving SEO and with the one that manages Google Analytics outcome were scheduled to receive immediate feedbacks and solve any open points.

3.2 Work Organization: Jira

3.2.1 Issue types

Jira is the software used by the company to plan, monitor and manage the project in an agile manner. It was developed by Atlassian and released in its first version

in 2002.
Jira Software allows users to view and manage their work in a transparent and flexible way, tracking and reporting activities and managing the software release processes. It is also highly customizable, since it can be personalized and adapted to suit different agile frameworks or methodologies, as Scrum, Kanban, Scrumban. Pursuing an agile approach is mostly facilitated by the introduction of *backlog* and *board* concepts in the software, since they help supporting team in everyday activities, showing the status of each work item, and allowing teams to closely monitor performance over time. In this sense, Jira facilitates the collaboration and ensures high control of the workflow, in turn increasing the productivity of the teams. Jira has proven to be efficient especially for projects managed fully remotely, as the one analyzed. Lastly, another advantage of Jira is its suitability for being integrated with a series of extensions and tools, also from third parties, to facilitate the coordination of the activities and to allow teams to give their best with minimum effort.
The examined project has been setup on Scrum framework, thus working on a Scrum board, with the aim of widespread Scrum approach within the company. The work items, which represent the building blocks of a project, must be included in Jira, in order to plan, develop and track the activities that must be accomplished. Bugs, requested features, and any other tasks you choose to track can all be considered *issues* in Jira. An issue in Jira is basically what it has been defined as *item* in Chapter 2, and those terms, as well as *ticket*, are used as interchangeable in this section. The list of all the tickets in Jira represent what has been previously defined as Product Backlog. The software offers, by default, a wide variety of issues types you can select to categorize the issues, and additional ones can be also added from the administration tool. An *issue type* consists of a short description (one or two words) of the topic which the issue relates, associated with a different symbol and color.
Of course, it is possible to deal with just a subset of them, depending on those that better suit the project. Again, customizing the issue types to match any method of project management preferred is also possible. In particular, the issue types used in software projects are often condensed to only four of the many available. The default proposal has been adopted also in the project under consideration, including *epics, tasks, user stories, bugs* and *sub-task*. In addition, it has been introduced also the *test* type. Labeling items with different issue types is useful especially for customizing the layout and behavior of each of them, as in

differentiating workflows.
The hierarchy among issue types is illustrated in Fig. 3.4. It emerges that epics are high-level issues, while task, user story and bugs, which are generally called *standard issue type*, are the smallest unit of an epic. They can be released within a single iteration complete and be immediately usable by the user. In turn, they can be divided into smaller chunks called sub-task issues. A more detailed description of each issue type is highlighted below.

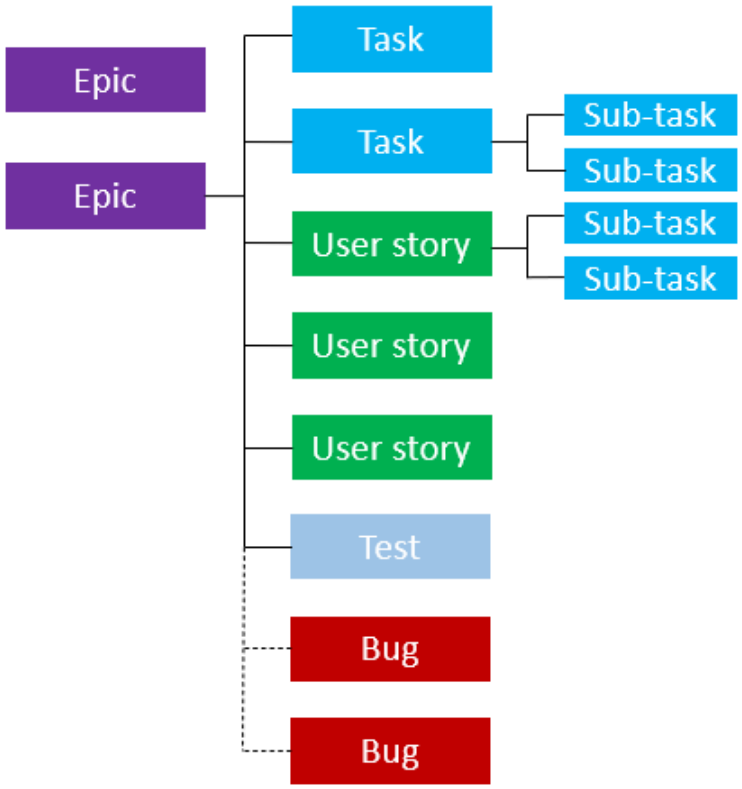


Figure 3.4: The hierarchy among issue types. The colours are those applied in Jira Software.

- An *epic* is the first level in the hierarchy. It represents a strategic, or long-term, initiative. It is a work activity too big to be carried out in a single sprint and then it needs to be gradually detailed and exploded, in order to broke it down into more stories and tasks. From a hierarchical perspective, thus, an epic can be considered as the *parent* of tasks, bugs, and stories. An explanatory representation of the epics is to consider them as a container of all the related activities. For software teams, an epic may represent a new feature they are developing.

- A *task* identifies any activity that does not directly impact the final user. For instance, uploading the spare parts catalog on the website is a task, since it does not affect the customer.

- A *user story* is an activity that has a direct value for the final user. It is a short and informal statement that describes a feature which adds value for a particular stakeholder, from whose perspective the user story is formulated. The expected value must be explicit, thus clarifying which is the target user, the requirement to meet and the reason why it should be satisfied. The upload of the spare parts entry point path in the header of the website is an example of user story.

- *Bugs* represent errors, flaws or faults in a system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. It is important to track bugs, in order to avoid compromising the success of future releases.

- The third level of the hierarchy is constituted by *sub-tasks*. They represent the action plan used to complete and implement stories and task. They are often very specific and technical, being useful especially if a standard issue requires multiple people working on it. Subtasks allow the team to have a detailed view of the ongoing activity for each item, ensuring a total transparency and increasing commitment.

- A *test* is an issue type, additional to the Jira default ones, that has been introduced in the examined project to mark activities that needs only to be tested. The only test issue reported in the project is related to test the *Back in stock flow*.

Epics used in the project

Taking into consideration the structure and the architecture of the website, together with the project goal, thirteen epics have been created for the specific project. Each epic refers to a specific component of the website to be developed and it includes all the items needed to fulfil it. To help understanding fully the project, a description of the epics is following.

- *Assortment* includes all the activities related with the catalog, in terms of spare parts, images, technical drawings, BOMs and their relationships.

- *Fulfilment* deals with the orders flow. It is the process of storing, packing, shipping orders and manage returns and exchanges.
- *Cutover* defines all the activities that must be done before the new implementations can be launched in production.
- There is a set of epics that involves the user experience (UX) and the user interface (UI). To distinguish the different web pages on which activities needed to be done, a further subdivision in more branches of the epic related with UX and UI has become necessary. Thus, *UX/UI Landing Page, UX/UI PLP, UX/UI PDP, UX/UI BOM page, UX/UI Footer* and *UX/UI Header* epics arose.
- *SEO*. This epic includes all the activities in charge of increasing the Search Engine Optimization of the website.
- *Google Analytics* contains the tasks to accomplish to meet the requirements issued by the GA team.
- *Checkout* involves all task related with the checkout, i.e., the final part of the customer journey where an order is -hopefully- issued.
- *FE Alignment on Master QA* is an epic that includes only the homonymous task. It defines an important activity that needed to be done before starting developments.

Although each epic may seem as an indipendent box, in most cases they have dependencies with the others that must be respected. The function in Jira that better shows epics, and their planning, is the *roadmap*. It represents a high-level programming which allows you to draw up the general picture and communicate the planning to the stakeholders. Once epics have been created, it is possible to schedule and link them in order to highlight any dependency.
As shown in the Gantt chart in Fig. 3.5, the considered project started with tasks and stories relates to *assortment* and *fulfilment*. This is because they can be implemented simultaneously and because the catalog and some parts of order flow were necessary for the development of the remaining project. Also, the *alignment of the front-end* with the master QA was necessary to be done first. Then, the project has been continued with *UX/UI* epics, in particular starting with the landing page and PLP, then PDP, BOM page, header, and footer. *SEO* epic could only begin after some implementation on the website have been done,

as well as the *checkout*.
Instead, *Google Analytics* items are developed last since they were not essential for going live. In fact, some issues were planned for being completed post-MVP. The only exception was, of course, when a development in the MVP was necessary.
Cutover is an important epic as it prepares the production environment in order to go live in a safe way. To plan thoroughly each activity to be carried out before launch, the cutover plan was made in advance taking into account due priorities and dependencies of the actions.

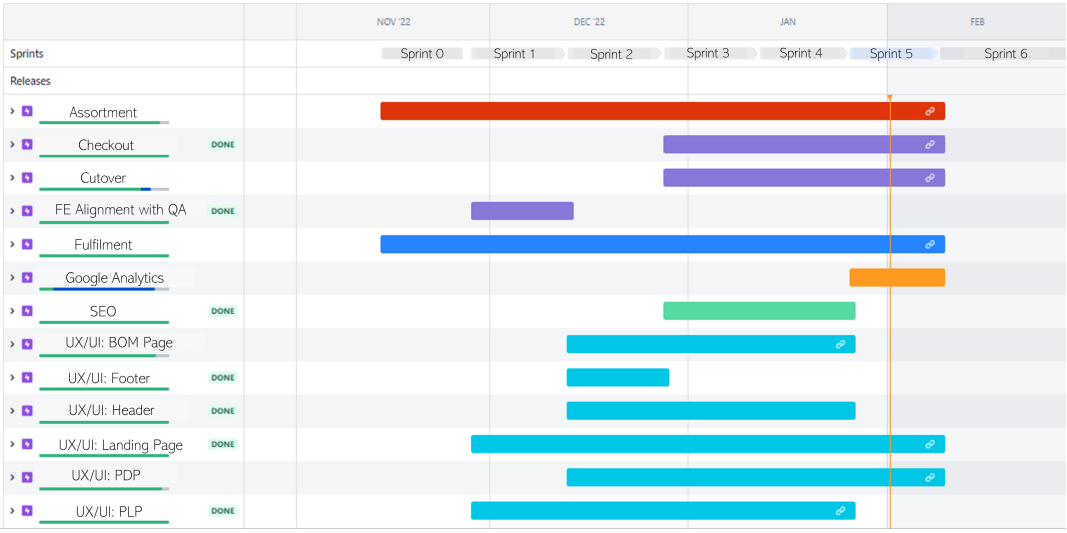


Figure 3.5: The Gantt chart for the project epics displayed in the roadmap function on Jira Software.

3.2.2 Issue Creation

When a new user story or task arise, they must be added in the product backlog. Creating a new *issue* (also called *ticket*) in Jira is simple. However, there are some criteria to be met to ensure order, accuracy, and consistency in the backlog. In this way, tickets are easily found and distinguishable, and any moments of confusion and waste of time are avoided. Thus, transparency is enhanced within the team members.
Looking at the project, it emerges that a ticket should fill the fields reported in Fig. 3.6.

- **Title**'s format: *[Website page] [Platform] Brief description.*
- **Description** of the requirement. It must be concise but comprehensive, to allow team members and stakeholders to get the most informed picture

Figure 3.6: An example of fields to be filled when a new issue is created on Jira.

when accessing it. It can be updated and refined along the project. User stories are described from a user's point of view, for example filling sentences as *"As a {type of user}, I want to {goal}, as then I can {reason}"*.

- **Business value** to highlight the value generated for the customers.
- **Acceptance criteria** that define conditions to be met to consider the story satisfied.
- The features that satisfy the **definition of ready**.
- Any possible **open points**.
- Any **relationship** with other issues.

Going more in detail, every ticket should also show the following information:

- **Assignee**, i.e., is the person in charge of the ticket in that moment.
- **Priority** of the ticket. This is essential to prioritize items in the backlog. The range of priorities varies among *lowest, low, high, highest* or *blocker*.
- **Sprint** associated.
- **Original estimate** in terms of time. **Time tracking** is an activity that would be good practice to develop but which has not yet been made mandatory in the project.
- **Platform** involved.
- **Labels**, if relevant.
- **Epic Link**.
- **Project** to which it belongs.

An example of issue details in Jira is reported in Fig. 3.7

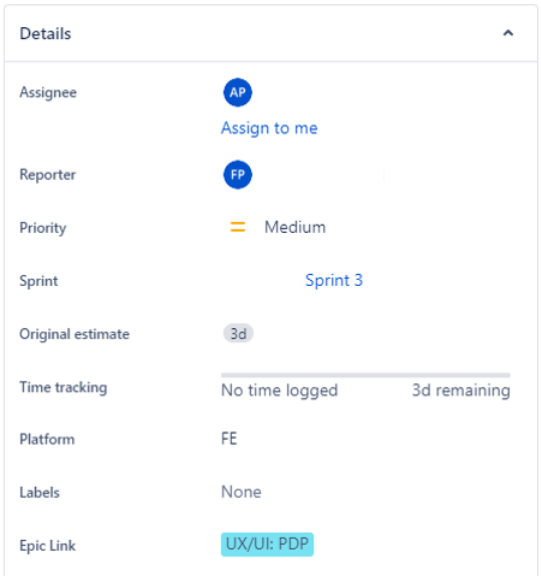


Figure 3.7: An example of Jira issue details.

3.2.3 Jira Workflow

As already mentioned, having different issue types is important to allow items to behave differently. A *workflow* is a scheme that portrays statuses and transitions between them. *Statuses* are the steps in the team's working process that describe the state of an activity. *Transitions* indicates how an item can be moved between statuses. Thus, for every issue type, a workflow is drawn with the aim to force the issues to move through specific statuses during their lifecycle. By removing ambiguity and uncertainty, a solid Jira workflow can increase productivity and efficiency for a single team. In fact, it clearly defines all the steps from which an item must pass before being considered finished. Having a single workflow model in Jira across different teams is the key point to increase homogenization among organization, to ease synchronization for shared developments and to promote an easier collaboration among developers and stakeholders.
As stated before, the main advantage for classifying items into different issue types is to customize their flow, creating status and transition *ad hoc* with regard to their needs.

Workflow for stories and tasks

Focusing on the project, workflows have been set differently for stories/tasks and bugs. The first one is more complex and detailed, because it covers the work process fully, from the creation of a new activity to its completion.

For tasks and user stories, the workflow applied in the considered project is shown in Fig. 3.8. All the tasks and stories have been added to the backlog in *to do*. This status is hence the starting point of the entire flow. Then, every single item is evaluated from a functional and technical perspective, and the work is estimated in terms of effort. In this period the item is set on *analysis*. Once the story/task has been studied and approved, thus the Definition of Ready satisfied, the item is *ready for sprint*, meaning that from this moment on, it can be moved from the product backlog to the sprint backlog. As soon as developers start working on the item, only after it has been allocated to the current sprint, the status passes to *in progress*. Here, considering the development of the task/story goes on without any problems, it is deployed in a Quality Assurance (QA) workspace and, therefore, it is ready to be tested. The item can be transferred to the *ready for testing* status. Hence, the tester can start *testing*. The tester must report the outcome of the test (KO or OK), the URLs object of the test and proofs of the validity of the test result. Furthermore, both mobile and desktop must be tested, attaching a proof. Then, multiple scenarios may arise. The worst one is the test fails (test KO). In this case, the item is moved to *bug-fixing* status, highlighting that some bugs need to be fixed before continuing the flow. On the other hand, if the development successfully works and the test is OK, the item moves toward the next status, that is *ready for release in QA* in pre-MPV and *ready for release in production* for post-MPV activities. QA environment is a workspace in which all developments are collected and tested without being live. The production environment is, instead, a workspace that will be released in production, i.e., on the website. Before being ready for the release or being closed, front-end and SEO items need an additional check to validate the work by the Product Owner, hence the item transits on a middle status called *PO review*. After the item is *released in production*, the homonymous status is set. Lastly, the item can be set in *done*, only after it has been successfully tested in QA or in production, satisfying the Definition of Done. A more detailed definition of Definition of Ready and Done is given in the next section.

However, some problems may occur, and the workflow should be able to deal also with those situations. The status of an item can be *on hold* if its natural progress is interrupted by something that is waiting to be solved. As you can see in Fig. 3.8, this status can be reached at any point in the workflow prior to be ready to be released.

In every moment a story or task can be also *canceled* if it is not needed anymore,

it has been duplicated or assigned to another team.

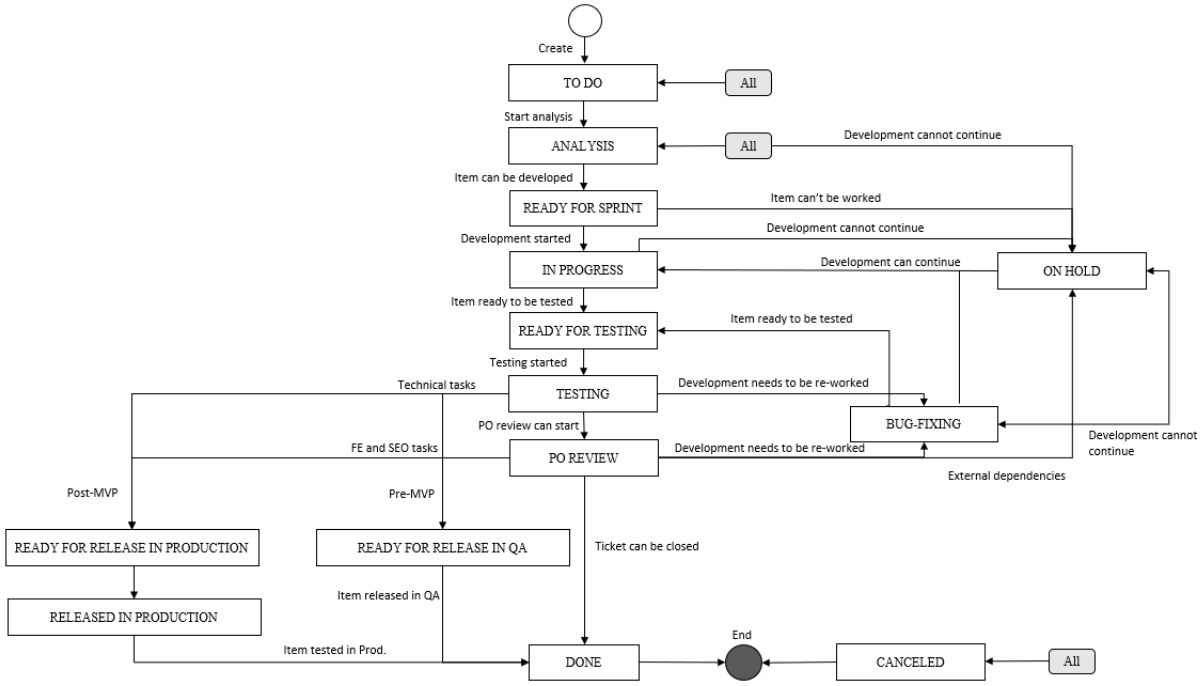


Figure 3.8: The Jira workflow implemented for user stories and tasks.

Workflow for bugs

The workflow is shortened for bugs. As Fig. 3.9 shows, when a bug comes out is set on *to do* and then it can be moved directly in *ready for sprint*. Hence, the analysis phase is skipped, and effort estimation is not tracked. This organizational choice relates with the fact that the peculiarity of a bug is its unpredictability. Therefore, spending time on its analysis is considered a waste, since a bug is something to be solved regardless and with a certain level of emergency, rather than an activity to be planned and evaluated in terms of feasibility. This decision was taken according with the business and developers, who argue it is difficult to estimate *a priori* the resolution time of a bug. On the other side, the lack of tracked bugs may cause some disadvantages in the project management. Indeed, even if at the end they are correctly solved, and this is the important thing, the amount of work that developers spend on bug-fixing remain unknown or approximate, distorting the results in terms of team performance and thus limiting optimal project management.

As well as for tasks and stories, also a bug covers *in progress* status until developers are working on it. Once the fix is completed, the item passes on *ready*

for testing and then *testing*. If everything is correct, the *PO review* is expected before being *ready for release in QA* (pre-MVP) or *in production* (post-MVP). During post-MVP period, the workflow also provides a status that marks that the fix has been *released in production*. Lastly, after an additional test in production to verify everything is properly working, the bug can be considered fixed and the status transit to *done*.
Unlike the one used for tasks and stories, this workflow does not provide the bug-fixing status, as this is the scope of the whole process. When some impediment is revealed and development cannot continue, the bug is moved to *on hold*.
Bugs can also be *canceled*, if their resolution is no longer within the scope of the project.

The transition from a status to another one, regardless if it is a story, a task or a bug, is obviously allowed only when a specific team member marks as successfully completed the previous one. For each status, the relative owner is summarized in Table 3.1.

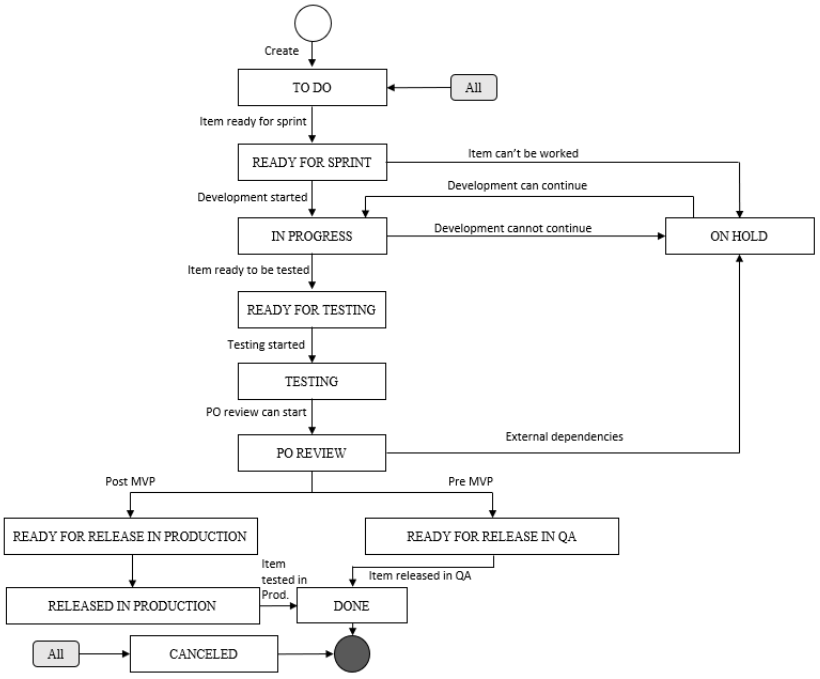


Figure 3.9: The Jira workflow implemented for bugs.

Definition of Ready

The Definition of Ready (DoR) consists of a set of conditions that must be met before a story, or a task, can be *ready for sprint*. First, the story must be func-

Status	Owner
To Do	BA
Analysis	BA/Dev
On hold	VTEX/Other
Ready for Sprint	Dev
In progress	Dev
Ready for testing	Tester
Testing	Tester
Bugfixing	Dev
PO review	PO
Ready for release in QA	Dev
Ready for release in prod	Dev
Released	Dev
Done	SM/BA
Canceled	SM/BA

Table 3.1: Issue status and related owner.

tionally and technically independent from the other stories as much as possible. Then, the size of the item must be small enough that it can be developed in a single sprint, and it can be broken down into tasks that can be estimated in hours.
With regard to the project, during the *analysis*, the item is first examined in terms of value preposition by the product owner or the business analyst. The description must be clear enough to allow the PO to understand how the product satisfies a customer need, its benefits and why it is better than similar products on the market. Then, the UX/UI designer has to check if the functionality already exists in the other brands or portals and has to complete the design in terms of UX and UI. Developers have to study the item in terms of its impact on the different platforms and identify possible dependencies with external teams. Furthermore, they have to estimate the item in *man-days* effort. Although it can be adjusted in the future, developers are required to estimate the most likely effort possible. Finally, the business analyst must clearly define the acceptance criteria and the tester must understand them.
Only when all those conditions are satisfied, the Definition of Ready is satisfied.

Definition of Done

In Chapter 2 the concept of Definition of Done (DoD) has been introduced. As provided by the Scrum Guide, the moment a Product Backlog item meets the

DoD, an increment is born. Practically speaking, it consists of a series of conditions that must be satisfied to consider an item in the sprint *done*.
In the project under investigation, the Definition of Done is different depending on the story.
An analysis-only user story can be marked as *done* if the analysis is completed and any open point is closed or tracked separately in a different story or task or subtask. Furthermore, any deliverable (as documentation or screenshots), if required, is included in the Jira ticket, and validated by the GIS PO.
During pre-MVP period, before a user story that requires development can be marked as *done*, it must be verified that the development is completed and tested in QA environment and that the product owner has finished the review of the development. Therefore, the complete product must be available in QA environment. Finally, it is ready to be released in production at the planned go live date.
The last scenario regards user stories that need development during the post-MVP period. To be marked as *done*, in addition to the requirements expected in pre-MVP period, the complete product must be also available and successfully tested in production environment. Furthermore, the involved persons or teams must be informed about the release in production.

3.2.4 Charts

In Jira it is also possible to create and personalize dashboards using filters. A *dashboard* is a useful tool that display gadgets which support the organization of the projects, allowing the tracking of the tickets and helping the team to respond accordingly. In the dashboard, several charts and graphs can be built, each offering different information.
At the beginning of every sprint, a dashboard was created to collect data about the project progress. Those deemed useful and mainly used in the examined project are the Burndown chart, the Workload Pie, the Sprint Health, and the Pie chart.

An example of the Burndown chart is displayed in Fig. 3.10 and it represents graphically the work left versus time. This gadget tracks the progress of the sprint, showing the remaining value to be developed, in terms of hours. Basing on the original time estimate, indeed, the Burndown consists of a gray line (*ideal work line*) which indicates the guideline for ideally performing the sprint and a

red one (*actual work line*) that represents the actual progress. The two work lines should be close as soon as possible, since if the actual is above the ideal one, it means that there is more work left than originally predicted and the project is running behind schedule. On the other hand, the project is moving ahead of schedule if the actual work line is below the ideal one, which indicates that there is less work left than originally anticipated. However, it can also be index of underestimated capacity. The Burndown chart and its peculiarities within the project are more deeply described in Sec. 5.1.1.

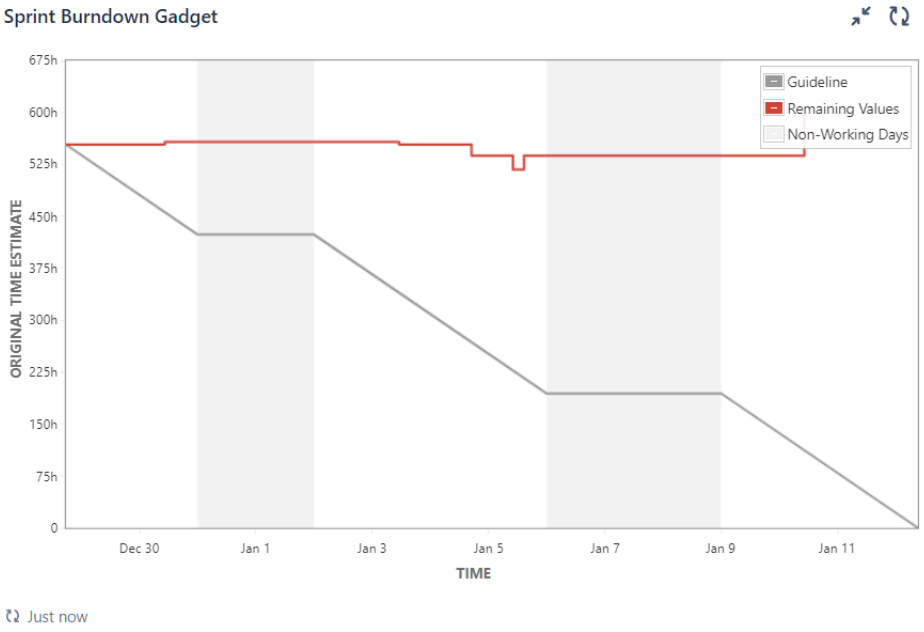


Figure 3.10: The Burndown Chart at the end of Sprint 3.

The Sprint Health (Fig. 3.11) provides a short and clear insight of the overall sprint progress. Here items that are completed, in progress or still to be done are summarized in terms of estimate effort. Some usefull insights also highlight the percentage of time elapsed, of work completed and of scope changed. Furthermore, the amount of blockers and flagged items is reported, too.

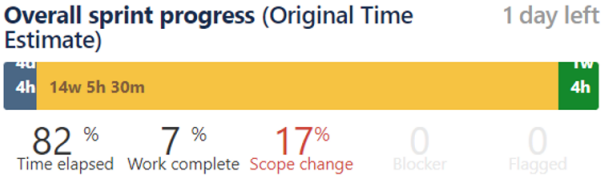


Figure 3.11: The Sprint Health at the end of Sprint 3.

Finally, the Pie Charts are mainly drawn setting *Status* as label. Indeed, they are useful to graphically show a recap of the status in which the issues of the sprint are. Simply looking at the graph in Fig. 3.12, it is possible to understand if most tickets is, for example, *done*, or *in progress* or still *in ready for sprint*. This graph was useful especially at the end of the sprints, when conclusions need to be drawn and results must be shared with the stakeholders. Furthermore, knowing the status of the issues at the end of the sprint is important to have a view of how many items need to be moved to the next sprint.

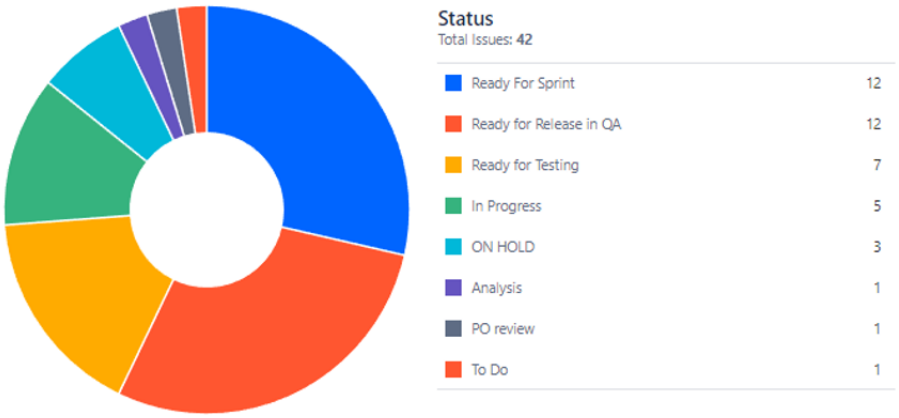


Figure 3.12: The Pie Chart at the beginning of Sprint 3.

The Pie Charts has been used also to track the status of tickets relates to a specific epic. Fig. 3.13 refers to UX/UI PDP epic and it is reported as an example.

Together with graphs to help keep track of ticket states and sprint progress, it is also useful to show the estimated effort allocated to each platform, as provided by the Workflow Pie (Fig. 3.14). Thanks to this chart, it is possible to see which is the platform with more work and to easily check if the amount of effort attributed is in line with the available capacity of the platform.

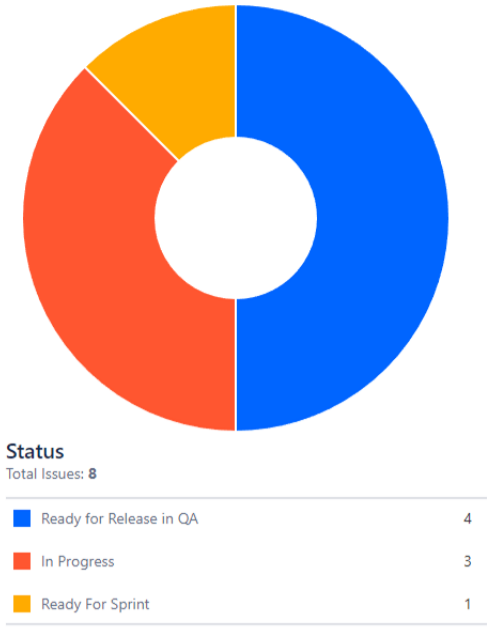


Figure 3.13: The Pie Chart of *UX/UI: PDP* epic at the beginning of Sprint 3.

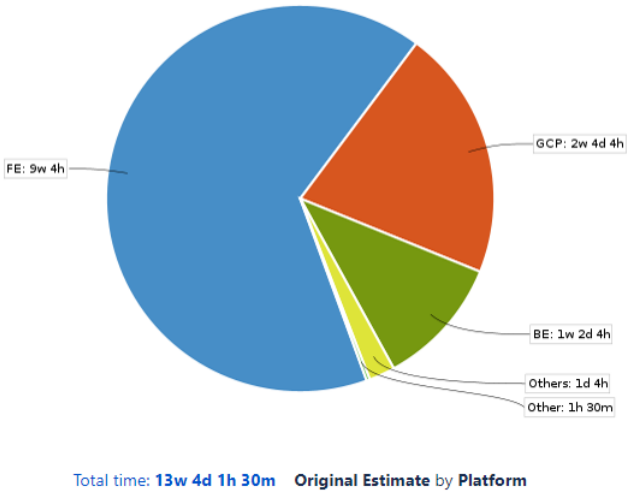


Figure 3.14: The Workload Pie at the beginning of Sprint 3.

Chapter 4

Scrum Application in the Project

4.1 Why Scrum?

As already mentioned in Chapter 2, selecting the most appropriate approach to manage a project could be not trivial. Choosing agile rather than traditional project management depends on many aspects, such as project features and organization characteristics. As instance, for a project that lends itself better to a plan-driven procedure, traditional is preferred, while agile is more suitable for projects that are planned and defined gradually [5]. As a result of further assessments, the domestic appliance company decided to privilige the Scrum approach. In this way, incremental and iterative development, frequent collaboration among team members and constant feedbacks from stakeholders were allowed. Deeply investigate on the strategic reasons that led the company to adopt Scrum is out of the scope of this thesis. Rather, the objective of this chapter is to analyze how Scrum approach was applied in the project of the case study and how its characteristics were enforced.

To be defined as a pure Scrum approach, all the statement in the Scrum Guide must be satisfied, as established by the authors. As a matter of facts, implementing Scrum precisely is very challenging and it leads organizations to carry out a Scrum project management that is customized rather than pure. This case also occurred for the analyzed organization. In fact, due to the nature of the project, it was needed to adapt and personalize some managerial aspects. Thus, looking at Chapter 2 and comparing the project concerned with it, there are some peculiarities that differ. For this reason, the project cannot be considered as a pure Scrum one. However, the general impression, as many organizational characteristics, is set within the Scrum framework, although not completely. Which are the

similarities of the project with the Scrum approach and what aspects, instead, differ are described in this chapter.

4.2 Scrum Elements

The diffusion of the agile mindset within the company officially occurs pursuing the Scrum framework. This approach is tentatively carried out even in the project under examination, as proved by the development in iterative cycle, thus the division in sprints, the organization of work into a backlog and the scheduling of Scrum Events. A more detailed explanation of how those aspects have been applied in the project follows in the next sections.

Sprint

As shown in Fig. 3.3, the management of the project includes sprints that last two weeks, i.e., ten working days. The choice of a such brief period is a double-edged sword. On the one hand, it allows a rigid management of the work in progress, within activities to be completed in short time intervals. Developers must follow a fast-working pace to meet the deadline and to be able to deliver within the end of the sprint. Operating under time pression may be a good way to stimulate employees to work hard, hence increasing the team performances. Consistently with the agile mindset, any waste of time is tried to be reduced. A shorter sprint may be the right manner to bring out and promptly solve any problems. Frequent comparisons with the business side help the team to drive the project toward the right direction. Indeed, in the Backlog Refinement, that occurred once or twice for week, feedbacks from stakeholders were collected and any proposal was discussed. Gathering feedbacks while the project is on-going is the milestone of an agile project management, since it avoids waste of time and effort on developing something that will prove not appreciated. On the other hand, recurrent sessions with stakeholders ran the risk of giving them the opportunity to change their minds repeatedly, overloading the team with new demands and ultimately leaving the project outside of its intended scope. In those circumstances, the PO and the GIS leader should be able to mitigate the introduction of requirements that were not involved in the project goals or unfeasible within the end of the project. Furthermore, shorter sprints need Scrum Events more frequently. This fact may lead developers to spend an excessive amount of time in meetings, rather than in developing. In fact, in addition to the daily update meeting, set to last half an

hour, the developers were engaged once every two weeks in the Sprint Review, which takes about 45 minutes of their time. Due to the necessity to finish the project in a tight time frame, it was avoided to subtract valuable time from the developers by organizing the Retrospective at the end of each sprint.
To sum up, it has been considered that two weeks were the most reasonable trade-off for the duration of a sprint.

Backlog

The concept of Backlog, proper of the Scrum framework, has been applied in the project. Since the project was fully remote, also the Backlog needed to be deployed virtually. This has been possible through the utilization of Jira Software. As mentioned in Chapter 3, Jira allows to collect all the activities needed to accomplish the project, creating the Backlog. These items could be put in the Product Backlog or moved to the Sprint Backlog, as provided by the Scrum approach. It is important to also state the conduction of the Backlog Refinement regularly. At the beginning of the project this action was usually bi-weekly because several characteristics should be still discussed. As the project progressed, the frequency decreased until the elimination. Hence, the considered project perfectly matches the Scrum framework within those Artifacts.

Daily Meeting

The first Scrum Event which has been introduced in the project is the Daily meeting. As already explained in Chapter 2, this meeting aims to inspect the work in progress, monitoring every item in the sprint. Focusing on the project, it was scheduled to last 30 minutes and it occurred every morning at the same hour, in order to reduce complexity. However, it could happen that the meeting ended early or late. People who constantly participate at the meeting were the core team, although it often happened that different guests were invited when their help was needed to solve a specific problem, as instance, the Vtex supporting team.
The main characters of the Daily Scrum were the developers and the Scrum Master. However, the participation of PO, BA and GIS leader was also essential to quickly make the point of the situation with the entire team, enhancing communication and proactivity. The SM was the person in charge of conducting the meeting so that it is useful to the progress of the project. Therefore, the SM

had to keep the participants focused on the purpose, so as not to waste time with unnecessary distractions. The Scrum Master had also to monitor the time to try to comply with the schedule. For their part, the attendees must commit themselves to actively participate in the discussion, giving the process fluidity.
The standard setting that has generally been adopted in every Daily meeting in the analyzed project can be outlined mainly in the steps described below. Notice that the frame of the meeting is completely up to the Scrum Master, who was free to change and adapt it in any moment. The meeting happened fully remote with the Scrum Master that was sharing the screen.

Step 1: Round table with developers

After every attendee has joined, the Scrum Master initializes the meeting through a round table with developers. In the meanwhile, the Jira board is shown by the SM. Developers are divided in three groups, depending on which part of the website they work on. There is the front-end, the back-end and GCP side. It is consulted one group at a time. Starting from one side, the involved developers had to inform the SM and the entire team about the ongoing activities, showing the Jira tickets they are facing. As agile approach provided, developers are completely autonomous in the choice of which tasks developing first. Of course, they have to follow the prioritization in the backlog. It is an occasion in which developers should express any doubts, misunderstandings, and inaccuracies about the items. Furthermore, any problems or obstacles that may impede the progress must be noted and analyzed.

Step 2: Tickets update

The Scrum Master must understand the information provided by developers and verifying that the work is going to bring a further increment toward the sprint goal. In particular, the SM must keep in mind both the overall progress of the sprint and the specific activities that concern it. Although everything should be tracked and written, knowing how to move within the sprint and the backlog, and what activities correspond to which platforms, may be important for the SM to keep up with updates and manage the project more easily. The Scrum Master must investigate on the effort of the activities mentioned by developers, thus if their estimation is still valid or needs to be updated. Another aspect to highlight and keep track of is the task's expected completion date. Make allowances for the collected information, the SM must update the Jira ticket, in terms of status,

priority, assignee or effort and adding comments to keep track of the advancement. If needed, the ticket's description may be revised too, with the addition of new details, relationships, or any further information.
The meeting has the scope to define the work planning for the day and share it within the team. Scheduling the Daily Event in the morning has advantages since the plan for the work of the day can be realized and directly executed.

Step 3: Open points and questions

When the round table and relative changes have been accomplished for FE, BE and GCP platform, the Scrum Master used to dedicate last minutes of the meeting to focus on any open points or questions. When time-consuming issues came up, the Daily meeting happened to last longer than expected. If on one side this allows to avoid wasting time by scheduling other meetings, on the other it runs the risk to keep some members busy more than necessary. Thus, it is always good that the Scrum Muster establishes the right trade-off between those two aspects.

Sprint Review

Focusing on the case study, the Scrum Master organized the Sprint Review once every two weeks. It is the Scrum Event which aims to inspect progress towards Sprint and Product Goal, showing the increment of value released in the sprint to the stakeholders (Sec. 2.4.4). Any person that could be potentially interested in looking at the progress of the project could join the meeting. Essentially, it is a meeting in which the core team update the local and business team on the work faced in the just-ended Sprint. The meeting lasted around 45 minutes, and it was recorded, since it could be consultable in any moment.
The standard Sprint Review meeting implemented in the analyzed project usually assumed the structure described below. It is schemed in Fig. 4.1.

Step 1: Sprint outcome

The first phase of the meeting concerns the outcome reached within the Sprint. After a brief overview of the project progress throughout the Gantt chart in Fig. 3.3, the Product Owner shows the results achieved in the sprint.
The sprint is analyzed in terms of *man-days* and Jira issues. Man-days indicates the working day equivalent if there was only one employee. Using this unit of measure, the total capacity available before the sprint started, and the net one, is

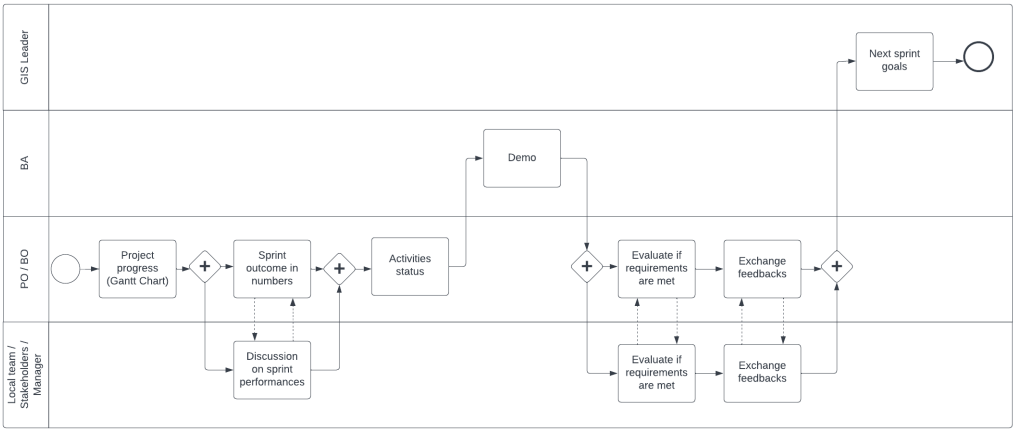


Figure 4.1: The Sprint Review meeting process.

shown. The *net capacity* takes into consideration a percentage of contingency for meetings, analysis and bug fixing. It has been estimated that the project needed an amount of contingency around 20% of its capacity for extra-developmental activities. Then, *planned* activities have been distinguished from the *unplanned* ones, i.e., those added while the sprint was already started. The number of canceled tickets is reported, too. Therefore, subtracting this number from the sum of planned and unplanned activities, the effective quantity of task is obtained. Lastly, *closed* activities, that are those considered developed, are highlighted. Notice that further to those released or done, also issues that still need to be tested or reviewed by the PO are included in this category. On the other hand, activities still in progress and moved to the next sprint are classified as *transferred*. For completeness an example of the table used in the sprint Review is given in Table 4.1.

At this point, it is possible to focus on the Key Performance Indicators (KPIs) obtained and comment them with the local team. Two KPIs are mainly used:

1. Capacity usage = MDs closed/ MDs potential.

The capacity usage indicator points out the closed activities over the potential capacity, in terms of man-days. A good capacity usage value should be around 1. A rate much lower than 1 means that the sprint was not as successful as expected, as it shows that there was more capacity available than actually used. On the other hand, this may indicate an initial overestimation of the capacity or a contingency too low compared to the effort employed in meetings, analysis and bugs fixing. On the other way around, a rate much higher than 1 denotes

	Definition	Mandays	Jira Issues
Potential	Sprint 1 net capacity	41	-
Planned	Activities planned before the beginning of the sprint (during the planning event)	32	13
Unplanned	Activities added to the sprint while it was ongoing	13	5
Canceled	Not needed anymore	1	1
Effective	Planned + unplanned - canceled	44	17
Closed	Activities developed (ready for testing, testing, PO review, released and done)	38	15
Transferred	Activities still in progress and moved to the next sprint (on hold included)	6	2

Table 4.1: Example of the summary of sprint results shown during a Review.

an underestimation of capacity at the beginning of the sprint or an inadequate choice of the contingency percentage.

2. Issue Performance = Issues closed/ Issues planned.

This KPI examinates sprint success throughout Jira tickets. Issues closed are compared with issues planned, giving rise to the issue performance rate. The considerations made before for the capacity usage indicator are still valid. A rate greater than 1 indicates that in the sprint there have been more closed tickets than those planned at the beginning. If it occurs, it means that the sprint has been successful, although attention must be paid to the accuracy of the data. Conversely, a low rate mirrors an unsatisfactory sprint since it was planned to accomplish more activities. Once again, the reasons behind can be multiple. Indeed, other than the fact that the developers may have been underperforming, it may also mean that unforeseen events have arisen, such as bugs to be fixed, which have proven to be longer than the estimated contingency.
Once the KPIs have been share to the local team, the Product Owner focuses on the status of the tickets at the end of the sprint. Throughout a pie chart as the one in Fig. 4.2, the percentage of tickets solved, as well as those still in progress

or in another status, are highlighted. The PO justifies to the local team the reasons why some issues still need to be completed and an end date of the activity (ETA) is assigned to them. However, developers or the entire team should not be blamed if the results does not match those expected but rather, a discussion for investigating the reasons may emerge.

Lastly, the Business Owner deeply illustrates the Jira issues faced in the sprint,

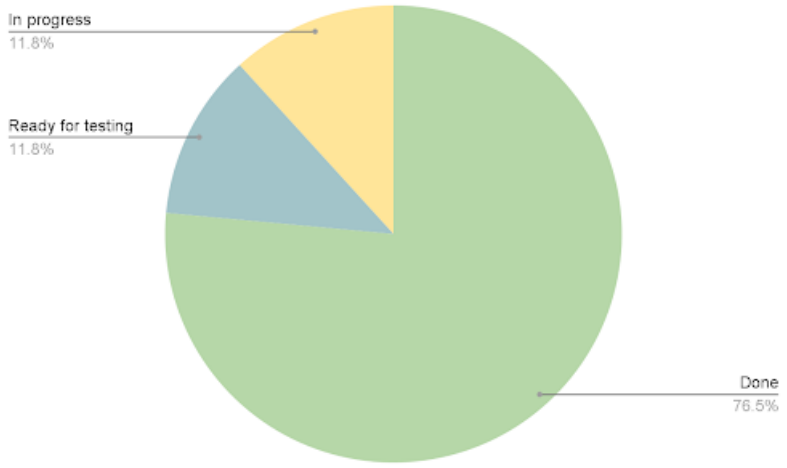


Figure 4.2: An example of tickets' status at the end of the sprint as reported during the Sprint Review.

with the help of a proper Jira dashboard created to measure the results of the specific sprint. Any open points are brought out and any comments considered.

Step 2: Demo

Subsequentially, the Sprint Review includes a demo phase. It is the most important phase since it proves the tangibility of the product. Due to the configuration of the project, the increments of value are released in production after the MVP is live. New tasks and features have been made available in a QA workspace over the course of the pre-MVP timeframe. It is crucial to demonstrate the releases to the local team during the Review meeting, even though it is not the live environment. Hence, the Business Analyst runs a demo to show the new developments and thus the potentially shippable increments of the sprint. In turn, the local team evaluates whether the requirements are met and provides feedback. If necessary, any adjustment is considered. The process to gather feedbacks and agree on changes while the project is going is the milestone of agile mindset. This allows the establishment of those fundamental iterations to ensure a lean project and gradual increments in value.

Once the MVP went live, the practice for the demo did not change, except for the environment in which it was carried out.

Step 3: Next sprint goal

Lastly, the Sprint Review meeting converges on looking at the next sprint goal. The items of the backlog planned to be done within the next sprint are shared with the local team. Furthermore, the expected increment is also highlighted. To be clearer, all the committed tasks are reported divided by their epic. Of course, the scope of the next sprint should also include the issue left incomplete in the previous one.
The GIS leader was used to provide this overview.

The organization is spreading the agile approach, and in particular the Scrum framework, throughout all the IT projects, as the one considered. In addition to the Scrum artifacts and events that were successfully put in place in the project, the Scrum methodology can be found also in the application of its values and in the general imprint that the management of the project takes. In fact, reducing everything that is superfluous and avoiding waste of time and effort are the two reference pillars that accompanied the team throughout the project. This was facilitated by the full adherence of the developers and the other members to the agile mindset. When a problem or a doubt arose, the most efficient form of disclosure was adopted, whether it was chatting or a scheduling a special meeting. No matter the processes and tools, the only purpose was to resolve the issue allowing the involved people to interact, without any spam or waste. Real time communication was preferred, as the more efficient and fast way to solve issues. Thus, lean communication is preferred over a more formal one, consistently with the agile philosophy described in Chapter 2. Furthermore, the Scrum values have been carried out. In fact, transparency, inspection and adaptation are concepts that were constantly present throughout the analyzed project. The implementation of these doctrines was eased by the recurrent events provided by Scrum. Transparency has emerged especially in the timeliness with which team members, and in particular developers, declared a problem, a misunderstanding, or a simple concern, with the purpose of solving it as soon as possible. Inspection was the milestone of the project. It especially occurred during the Daily meeting and through the monitor of KPIs and charts. Adaptation has been also a key concept

within the project. As instance, it happened every time a change in the planning was needed. The team proved to be perfectly able to deal with it. Receiving continuous feedbacks from stakeholders is another aspect typical of agile. It is strictly connected with the concept of iteration and incremental addition of value. Finally, the project saw the implementation of the five basic values provided in the Scrum Guide, too. In a nutshell, they are commitment on the goal, focus on the task, openness about the work, respect for the other and courage to take actions.

4.3 Deviations from Scrum

However, some features of the project make it difficult -if not impossible- to purely apply Scrum project management. Deeply analyzing how the project under consideration was handled, some aspects that differ from Scrum can be highlighted. They are reported within this paragraph.
Since the one under consideration was a building project, it was impossible to release a deliverable in production at the end of each sprint. New functions and codes were developed iteratively within almost every sprint but released only in the QA environment. The increments to which the Scrum approach refers in the Guide were thus delivered in a testing environment. The passage from the test environment into a production one was scheduled for a specific day, that is the go live data. This is the first aspect that does not perfectly match a pure Scrum approach, because it usually involves constant releases that can immediately go in the customer's hands. However, business and local teams, that were those in charge of establishing the requirements, were informed about the new tasks in the Sprint Reviews, and they could already have a look at how the output would come.
Among the Scrum Events implemented in the project, the Sprint Planning has been customized. As explained in Chapter 2, it should be done before a new sprint starts with the purpose to take the most prioritized items in the backlog as targets for the new iteration. However, in this case the Scrum Master planned every sprint in advance. After all the items in the backlog have been estimated in terms of effort and priority, they were assigned to the pre-MVP sprints. The aim was to fill the sprint capacity with tasks and stories, paying attention to any potential relationship among the activities and to the contingency effort planned for bug-fixing and not deployment actions. In this way, a provisional

program was established, and the project could be better managed with a long-time perspective. Nevertheless, a proper meeting was regularly taken at the end of every sprint in order to verify the plan for the next one and make any adjustments. Since the plan was temporary, the necessity to add, move and reschedule activities or other modifications occurred frequently. The choice was pushed by the project leader, that wanted to allocate the right number of developers to complete the project on time. The support of Jira Software was essential in this phase, since it allows to move easily tickets within sprints and, through charts, propones an overview of the estimated capacity. Another aspect emerged in the project that is not provided by a pure Scrum management is adding activities to sprint while it has already started. This is an indication of poor planning, either because available capacity has been underestimated or because the estimated effort of the activity has proved to be excessive. On the other hand, it increases the challenge of the project, engaging skills for the team members as adaptability and flexibility.
The Sprint Retrospective is another Scrum event that was not realized within the considered project. The reason was the lack of time. It should have been done once every two weeks, that is the duration of the sprint. Since it was estimate that the project time was tight and spending much time in meeting could have been risky to go live with the MVP, it has been agreed to remove the Retrospective ceremony. If one side precious time was dedicated on developing tasks and getting closer to the goal, on the other hand a moment dedicated to the discussion of the internal process was missing. Although the team was broken in with agile, having an opportunity to improve the working organization could be always a good point. Indeed, it eases the refinement of the Scrum approach, the enhancement of the performances and, thus, the increment of satisfaction. However, at the end of the previous project of spare parts integration, the Scrum Master involved the team in a Retrospective meeting in which positive and negative aspects of the overall project management were discussed. In this occasion, aspects as what went well and what could have been improved emerged and they turned out to be useful insight for the following project, that is the one analyzed in this thesis, given that the team members for the project remained mainly the same.
About the team composition (Chapter 3), the project resulted to be overloaded. As previously said, a Scrum team should be composed by a Product Owner, a Scrum Master and some developers. However, the presence of both a PO and a BO was sometimes misleading because they owned the same project but relying on different departments. In substance, they often were in charge of accomplish

the same tasks. The risk here was to obstruct the process, slowing down the flow of information and activities.
Other minor project peculiarities to point out that slightly vary from Scrum are the implementation of a sprint 0, the longer duration of the last sprint and the different meetings length. The term sprint 0 refers to a period, before the first sprint officially started, during which a set of activities have been completed. The tasks and stories have been done at the best effort, that means in the odd moments of developers. The specific activities were tracked in Jira as in a usual iteration, but Scrum Events did not take place in this time span. As shown in Fig. 3.3, the sprint 6, that covered the hypercare period, lasted three weeks instead of two. Even if Scrum approach should include sprints equally long, sprint 6 is an exception because this time was dedicated only to solve potential bugs and enhancements after MVP period. Finally, scheduled time for the duration of Scrum Events, reported in the Guide, are different from those selected in this specific project. Although the ones listed above are characteristics that somehow deviate from Scrum in a less relevant manner, it was assumed that for completeness their mentioning is also important.

Chapter 5

Results

5.1 Team Performance

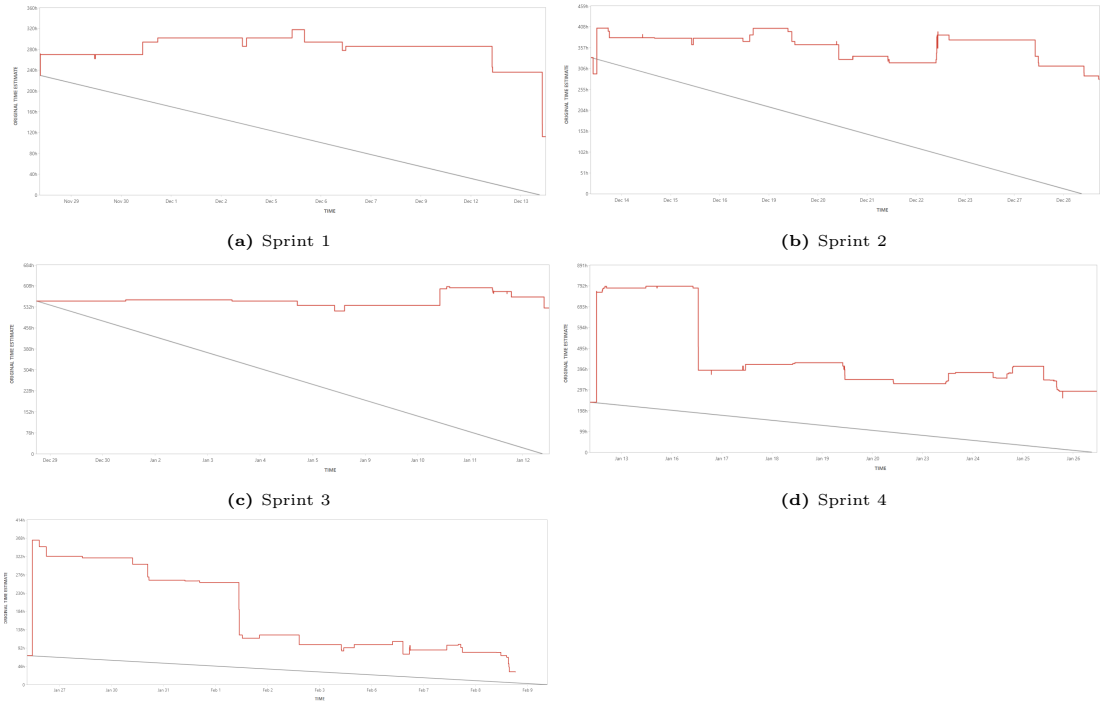
Technically speaking the project was ready to be delivered on time, since all the necessary task planned for the pre-MVP was successfully developed. The scope of this paragraph is to evaluate the outcome of the project, looking at the technical performances of the team and investigating how they are related with the Scrum approach. It was done through an interview to the core team. Furthermore, the advantages and disadvantages of the Scrum that emerge are described. Finally, the enhancements and takeaways that can be brought to the company are investigated.
Starting with the study of the team productivity, it is important to state that there are different metrics that may help the agile team to monitor their performances. It is essential to clearly understand what the meaning of the different charts is, finding out valuable information to streamline the project, manage its progress and respond accordingly.
Although the project counted six sprints, this analysis takes into consideration the pre-MVP ones (i.e., from sprint 1 to 5), since most of developments needed to be done during this period. Indeed, the hypercare period, that covered sprint 6, focused only on bugs fixing and small enhancements.

5.1.1 Burndown Chart

As anticipated in Chapter 3, the Burndown chart provides a comparison between the total work remaining in a sprint (red line) and the estimated trend that should be followed to achieve the sprint goal (gray line). It also allows to understand

the likelihood that the team has in order to complete all the tasks on time and to reach the goal. Knowing it, the team can investigate on issues and act promptly to stay on track. This chart can be easily extracted from Jira Software. The horizontal x-axis indicates time, whereas the vertical y-axis indicates the amount of work that needs to be carried out within the end of the sprint. Y-axis may be built using several estimate statistics, depending on which aspect is better to highlight. As instance, it might show the number of tickets, the capacity, or the original time estimate.
Although Burndown charts are usually used to monitor the work in progress and to fix eventual issues on time, having a look on them may be worthwhile also *a posteriori*. Analyzing them can be helpful especially to draw conclusions on how the overall project went. In fact, they offer important insights on the ability that the team has shown in successfully completing the sprint goals. Furthermore, it allows a comparison on different sprints. Combining this information may be critical for the team, mainly for PO and Scrum Master, to understand the causes of potential failures, learning from mistakes and improve them. This logic goes hand in hand with the agile philosophy concerning adaptability, fast-learning and continuous enhancements.
Focusing on the examined project, Burndown charts for each sprint are reported in Fig. 5.1. Here, some interesting observations arise.

As can be seen for all the sprints, the pattern does not decrease linearly, as it is ideally expected. The trend is rather irregular. It is mostly straight, with sudden changes of directions. A line falling downward means that works has been released, thus marked as done. The upward bars represent an increase on the estimated effort or the addition of new activities to the sprint. According to the charts, it appears that the sprint goal has not been reached in sprint 2 and 3, whereas it has been steadily achieved in sprint 1 and sprint 4. The reason behind is a delay in the releases (in QA) scheduled for sprint 2 and 3, as it is shown also by the CFD in Fig. 5.3. Although most of the tickets were already developed, the Burndown chart considers them as accomplished only when they are set on the *done* status. In this case, some tickets were still pending among the sprints, because incomplete. However, given the project organization and go-live date, releasing new functionalities in QA within the end of the sprint was not considered a high priority. Thus, the Burndown chart outcome must be contextualized and managed carefully, otherwise it might run the risk to be misleading, at the expense of the efficiency.



(a) Sprint 1

(b) Sprint 2

(c) Sprint 3

(d) Sprint 4

(e) Sprint 5

Figure 5.1: Burndown Charts

As illustrated in Fig. 5.1d and Fig. 5.1e, the trend in sprint 4 and 5 is quite different from those in the previous sprints. Indeed, with the approach of the go-live date, even more and more tasks have been completed as indispensable for the launch in production. Therefore, the remaining work started decreasing, as reflected by the downward trend of the lines in the charts. The little amount of work left consisted of minor changes and enhancements that was planned to do during the *hypercare* period.

5.1.2 Velocity Chart

The Velocity chart is a bar graph that displays the amount of completed work (green bar) versus those planned at the beginning of each sprint. It is helpful to predict how quickly the team can work through the backlog because the report tracks the forecasted and completed work over several sprints. The last sprints are reported on the x-axis, whereas the y-axis displays the estimation statistic. In Jira software, they can be based on story points, original time, issue count or customized. The commitment (gray bar) is calculated basing on the estimation of all issues in the sprint when it begins, thus without considering any changes and added activities while the sprint was ongoing. The green bar shows the

completed estimates at the end of the sprint, including also any changes made after the sprint started. Therefore, it is possible to compute the team's average velocity. This value can be used to predict how much work can be completed by the team in a future sprint. It becomes more accurate and reliable over time, as more data become available, and the team gets better at estimating issue.
The Velocity chart performed by the team in the project under examination is reported in Fig. 5.2, with the original time set as the estimation statistic. The sprint 0 is not taken into consideration given that it has been played at the best effort. It clearly emerges that the team executed pretty good in sprint 1 and 2, since the completed amount of work is similar to the committed one. However, the team was not able to accomplish all the work committed at the beginning of sprint 3. The reasons are multiple, originating from both agile limitations and underperforming team. On one hand, the committed work at the beginning of sprint 3 turns out to be much higher than those in sprint 1 and 2. This fact is due both to an incorrect planning and to a shift of incomplete issues from sprint 2 to sprint 3, resulting in an increase in commitment. This bad result comes from a limitation on the way the project has been managed. On the other hand, the team itself performed worse, as confirmed also from other indicators (reference al grafico dei KPI's). However, the team has proven to be highly performing in sprint 4 and 5, accomplishing more work than the one planned at the beginning. Many activities were added while the sprint was on-going, giving the approaching go-live. It also includes the completeness of activities left over in the previous sprint. This outcome is consistent with what emerged in the Burndown and Burnup charts analysis.

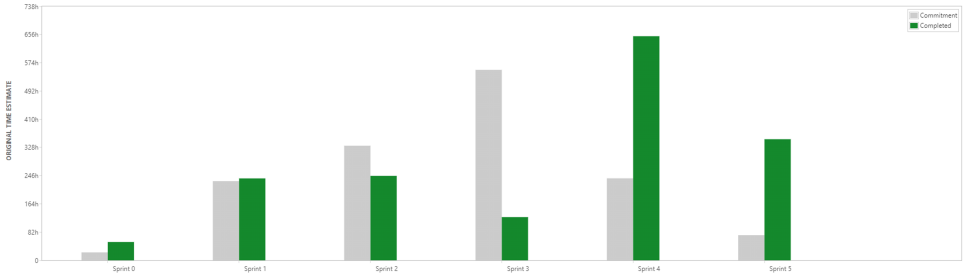


Figure 5.2: Velocity Chart.

5.1.3 Cumulative Flow Diagram

The Cumulative Flow Diagram (CFD) is an area chart that displays the various statuses of the work items within a timeframe. It is automatically built by Jira,

that marks each status of the workflow with a different color. The time is located on the x-axis, whereas the y-axis features the number of issues. The CFD offers a wide view of the project progress, permitting to identify, already at the first glance, the prevailing status in which the project is. The CFD counts the quantity of issues that every day pass through each column of the board, i.e., the workflow statuses. Cumulative means that if an issue is moved from *in progress* to *done*, the chart accumulates 1 for both statuses in that day. The CFD is also used to identify bottlenecks. Indeed, an area that is widening vertically over time probably points out the presence of an obstacle and needs to be further investigated.
The CFD in Fig. 5.3 shows how the analyzed project has advanced over time, considering it from the beginning of the sprint 1. Keeping tracks of how the issues of the project passed through each column of the Scrum board is essential to discover on which status they mostly accumulated. Having the view of the issues situation on the entire project, it is possible to state that it was successfully performed by the team, since all the necessary tasks were successfully completed before the go-live, with no significative obstacles. As expected, the graph shows a growing trend in tasks marked as done. As the go-live date approaches, closed issues increase rapidly. Going deeper, three main bottlenecks can be identified in the diagram but none of them constitute a blocking point for the project. The graph, indeed, depicts a blue area that started to expand vertically in late December and continued for about three weeks. This period corresponds to sprint 2 and 3. Hence, the first congestion refers to the *ready for release in QA* column. Several tasks were stuck in this state for days, until the obstacle was finally overcome, and the releases done. The *ready for testing* status revealed to be another congestion phase. It seems there was a delay in making the tests during January. A change in the allocated resources and holidays are accountable for this. Lastly, during the third week of January (sprint 4), several issues accumulated on *in progress* status. This is consistent with the insertion of new tasks added before the go-live date. However, the mentioned bottlenecks have not brought significant delays or serious consequences for the project, but rather a shift of the releases in QA. Although the issues were allocated to be completed within the sprint, this was not always verified, due to some difficulties, such as those highlighted above. However, given that these obstacles still occurred in the pre-MVP period, not releasing at the end of every sprint was considered acceptable as well.

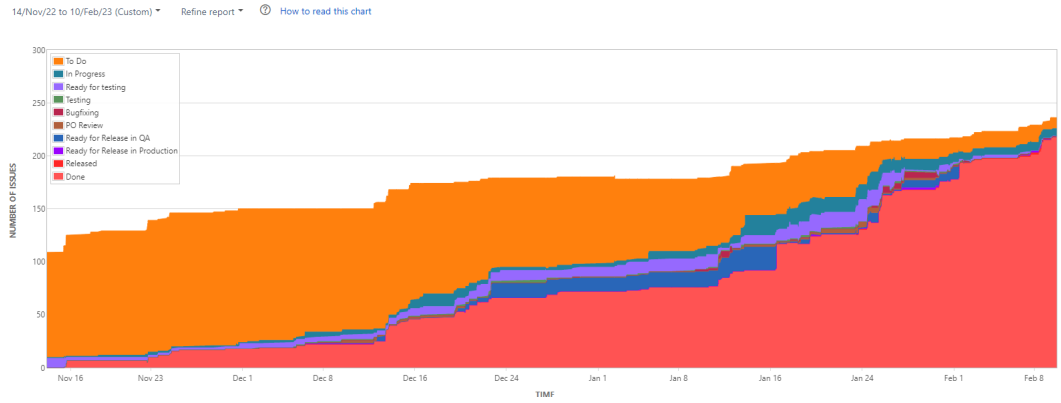


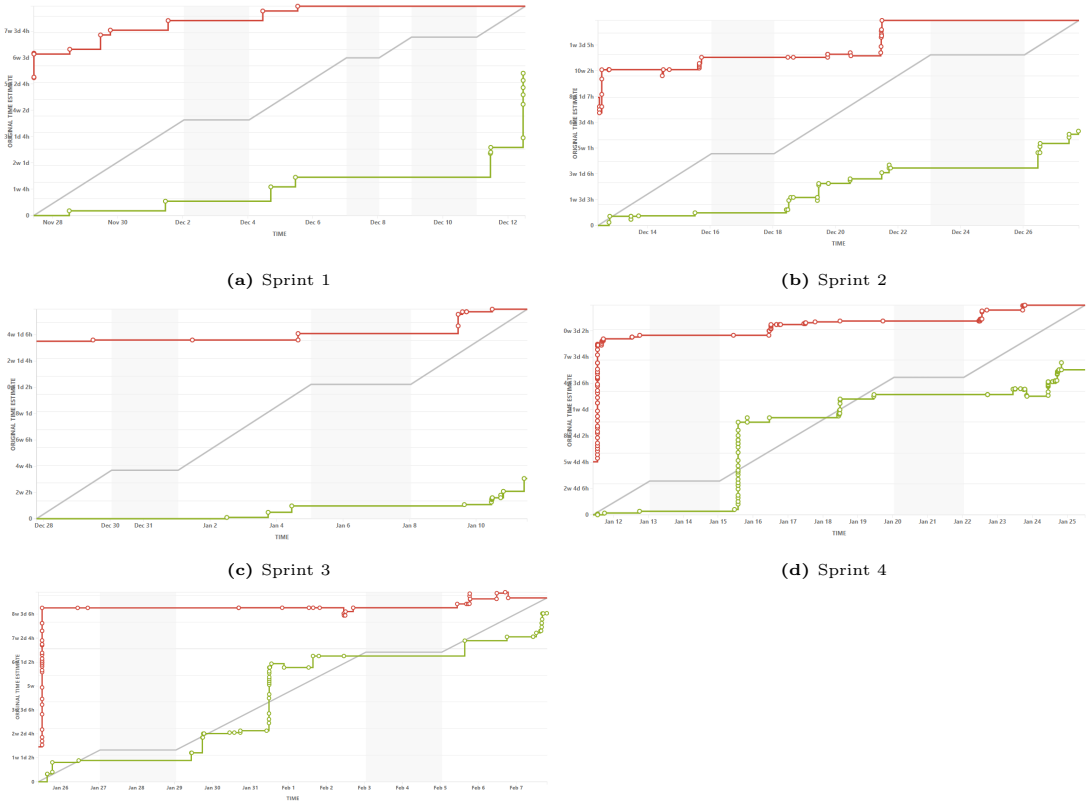
Figure 5.3: Cumulative Flow Diagram.

5.1.4 Burnup Charts

Another important metrics mainly used in project managed in agile is the Burnup chart. It provides a visual representation of the sprint's completed work compared with its total scope on a day-to-day basis. Differently from the Velocity chart, the Burnup chart is updated as soon as a change in the sprint scope occurs, even if it happens while the sprint is already started. Hence, thanks to this report, it is easy to pinpoint problems like scope creep or project path deviation. Again, the x-axis represents the time, generally in days, weeks, or sprints. The unit of work is shown on the y-axis. It can be reported in terms of story points, issue count, original estimated effort, and many others.

Jira provides the Burnup chart for individual sprints in the project. As represented in Fig. 5.4, the red line depicts the total amount of work necessary to complete the goal. The green line represents, instead, the completed work, thus the progress of the project. The sprint goal is achieved when the two lines meet. Therefore, the distance between the lines represents the residual work that still needs to be done. To have a clearer view, Jira includes in the Burnup also a guideline (the gray line) that is the ideal path the team should follow to reach gradually the goal. The red circles point out a change in the scope, as the inclusion of a new task, whereas the green ones state the activities completion.

According with the Burnup charts of the specific project of this thesis (Fig. 5.4), they confirm the project trend obtained analyzing the other metrics. Once again, it emerges that sprints 1, 4 and 5 are the sprints that the team performed better, with the goal almost completely achieved. However, the ideal guideline is not perfectly matched in any case. The sprint 1 features a spread of completed work during its last day. It means that all the statuses in the workflow have been successfully passed in time for most of the tasks, and releases (in QA) made by the



(a) Sprint 1

(b) Sprint 2

(c) Sprint 3

(d) Sprint 4

(e) Sprint 5

Figure 5.4: Burnup Charts

end of the sprint. The progress of the work done in sprint 4 is for some stretches even higher than that idealized by the guidelines. The reason is the releases of several developments in QA and therefore their definitive completion. They correspond to a peak in the graph. With regard sprint 5, the trend is very satisfactory since the work performed reached the scope quite gradually. In particular, it appears there were two main moments in which most of the releases occurred, one at the beginning and one at the end of the sprint. Looking at the sprint 2, the trend is quite regular and characterized by frequent releases. Despite this, the planned work has not been fully completed, owing also to a huge addition of the quantity of work when it had already passed more than half sprint. Team's ability to achieve the goal in sprint 3 appears to be very low. Indeed, the chart shows little work completed, except during the last days of the sprint, ending far away from achieving the sprint goal. On the other way around, the committed work was too high compared with the capacity.

5.1.5 Issue Performance and Capacity Usage

In Chapter 4 it has been introduced the concept of the two KPIs that were used during the project to compute the team's efficiency in the sprint. Those indicators were considered so important that they were shared in the Sprint Review as metrics for the team performance. The graph in Fig. 5.5 has been built to illustrate and compare how the sprints have been performed in terms of these KPIs. On the x-axis the rate is reported, whereas every single sprint states on the y-axis. In a nutshell, the Issue Performance indicator represents the ability of the team to develop new functionalities. It is computed by the number of issues *closed* over the issue planned during the sprint planning. Notice that issues considered *closed* are not strictly those in the *done* status. Indeed, tasks and stories were set as closed if their development have been completed, nevertheless they still need to be tested, reviewed, or released. To make the logic simple, the issues are considered closed if the effort needed by developers to deploy them is not transferred in the next sprint. Looking at the graph, this rate results to be high in every sprint, as it is always around 1. It means that developers were able to widely deploy the committed tasks and stories, even if considering them as officially closed on Jira required further steps. Moreover, it emerges that the developments exceeded those planned in sprint 1, 2, 4 and 5, reflecting the team's capability to adapt and respond successfully to changes. In sprint 3 the Issue Performance was lower but still high, with a team able to develop most of the tasks among those planned.
The Capacity Usage measures the performance of the team in terms of effort. It is based on the ratio between amount of work closed and potential capacity available at the beginning of the sprint, in terms of *man-days*. The Capacity Usage is important mainly for two reasons. On one side, it allows to understand if the planning of the sprint was coherent with the available capacity. On the other side, it is helpful to set an adequate contingency for meeting, bug fixing or if something come up. The graph in Fig. 5.5 highlights values of the KPI around 1 for all the sprints, except two falls in sprint 3 and 5. This event is due either to a bad estimation of the capacity contingency and to an excessive capacity at disposal. Due to the go-live, most of the tasks in sprint 5 were already completed, thus the team capacity was mainly dedicated to fix bugs and contingency. The peak in sprint 4 states a great value of the Capacity Usage, demonstrating that the team was able to successfully accomplish unforeseen activities within the same time horizon.

The high scores reported in the graph mirror well-performed sprints. Indeed, both the KPIs are around or higher than 1, meaning that the developers usually highly performed or even overperformed. The Issue Performance indicator line is always above the Usage Capacity, although the trend is similar for both the KPIs. Exceptions are two points where the lines diverge. First, the blue line forms a peak in sprint 2 that does not show up in the orange line. Then, the score of the Capacity Usage in sprint 5 is far from that reached by Issue Performance. In those cases, the performance in terms of issues is largely greater than the one computed considering *man-days*. This can be due to a misleading estimation of potential capacity devoted to the contingency and to a large addition of tickets while the sprint was underway. In conclusion, by assessing the overall project

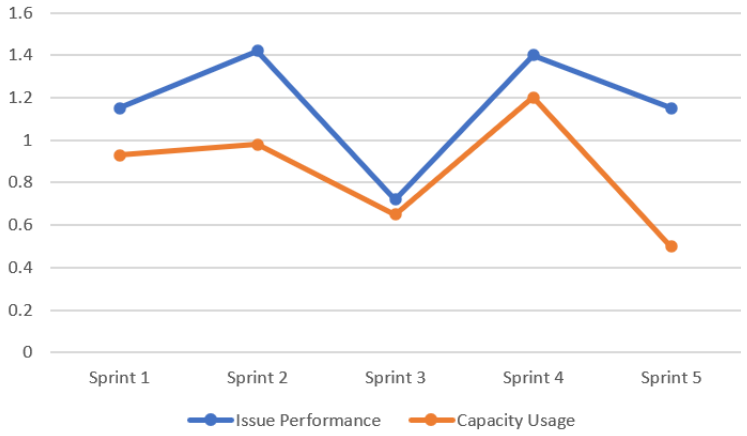


Figure 5.5: Cumulative Flow Diagram.

trend it emerges that it has been performed with success. The project goal has been achieved and the website was technically ready to be delivered on time, as confirmed by the CFD (Fig. 5.3). Despite many ongoing requests by the stakeholders to make product changes, to introduce new requirements and evaluate new functions, the team has proven that they have managed changes adequately and efficiently. In terms of technical productivity, the team was able to develop tasks and stories with effectiveness in all the sprints, as reported in the KPIs' graph (Fig. 5.5). Comparing different graphs, it is important to keep in mind how the different metrics consider the activities, whether as *completed* or *done*, to get the right information. Investigating the project trend more deeply, sprints 1, 4 and 5 are those that were better executed, with Burndown, Velocity and Burnup charts consistent. The sprint 2 and 3 have a lower performance for the reasons already mentioned.

5.2 Interview Outcome

Until now, the efficiency of the project and the team performance were discussed basing on empirical data. To better understand and investigate their relationship with the manner the project has been managed, a qualitative interview to the core team was done. Each member was asked to answer specific questions through an anonymous questionnaire. It was shared by email and provided by Google Form. The purpose of the survey was to find out how the Scrum approach has been perceived by the team. Out of fourteen members, eleven answered the questionnaire.
The questionnaire can be divided in three sections. It started with general questions to understand how informed the team about the Scrum approach is, what is the level of confidence and experience the team members have about it. Then, it has been asked to assess the project in terms of specific key dimensions in order to analyze its efficiency. The last section focuses properly on the way the project has been handled. It aims to collect information on how the team has perceived the Scrum framework, which advantages and disadvantage they encountered and the overall satisfaction.

5.2.1 Section 1: Team Experience

The goal of the first section is to understand if team members have already dealt with the Scrum approach during their job, thus if they were already skilled or if they were beginner.
It appears that all the members of the team have already experienced with Scrum in earlier projects. As shown in Fig. 5.6, 27,3% of the respondents worked in Scrum in one or two previous projects, almost the half of members experience from three to five projects earlier whereas 18,2% from five to ten. Finally, a professional experience in more than ten Scrum project is covered by 9,1% of the respondents. Furthermore, respondents appear to be quite skilled on Scrum project management, declaring to know it *fairly* (72,7%) and *sufficiently* (27,3%). Those data depict a team that is quite familiar with Scrum. A team that has already encounter the Scrum approach is a strong point for the organization as it is facilitates the adoption of the framework and helps to embrace the agile mindset faster. Indeed, be used to work in an agile setting is essential to rapidly comprehend the organizational process and establish a quick and efficient routine.

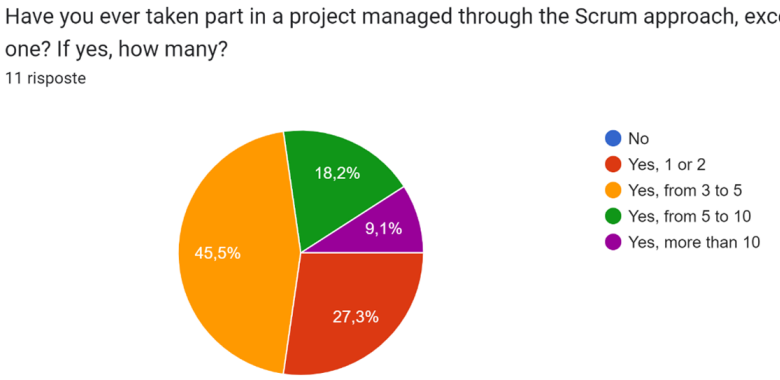


Figure 5.6: Experience in Scrum.

5.2.2 Section 2: Project Evaluation

As anticipated, the second part of the questionnaire dealt with the evaluation of the project, with the aim to investigate how the project organization results to be effective. The respondents assessed the project in terms of five dimensions, assigning a score from 1 (*ineffective*) to 5 (*very effective*). They are:

- Ease of keeping up with the activities.
- Clarity in the activities to be carried out.
- Ability to involve you in the project.
- Ability to involve you in the team.
- Efficacy of communication within the team.

As illustrated in Fig. 5.7, high scores were attributed to all the considered features, confirming the project as overall *effective*. Indeed, the answers involved only right values (score higher than 3), without any *ineffective* or *little effective* results. In terms of easiness of keeping up with the activities, the project has been evaluated as *effective* by most of respondents (73%) and *sufficiently effective* by 9%, whereas 18% assessed it as *strongly effective*. Similar percentage resulted also for the level of involvement both in the project and in the team. Therefore, the project has been perceived as prevalently *effective* in terms of teams' engagement and interest in reaching the goal. With regard the clarity in the activities to be carried out, the effectiveness of the project resulted to be lower, even if still dominant. In particular, 64% of respondents stated this dimension as *effective*,

whereas the remaining ones split between *sufficiently effective* (18%) and *strongly effective* (18%). The last dimension, i.e., the degree of communication within the team, is still prevailing effective. The respondents scored it as *sufficiently effective* (9%), *effective* (82%), *strongly effective* (9%).

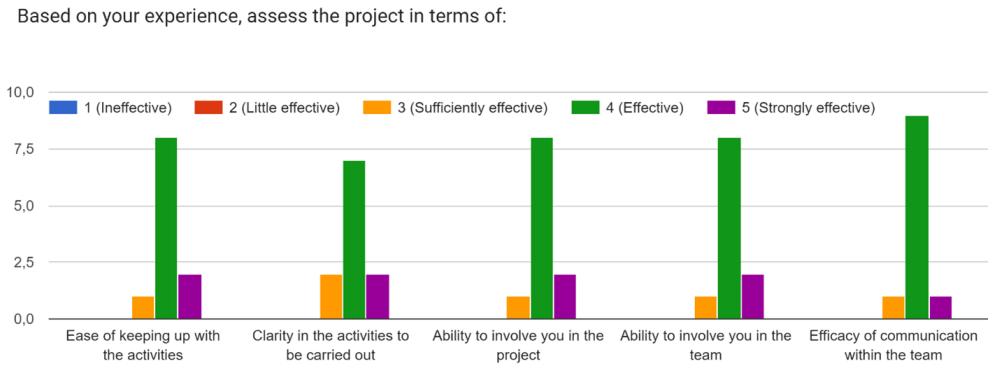


Figure 5.7: Project evaluation.

5.2.3 Section 3: Scrum Relationship

The main part of the questionnaire aims to investigate what is the estimation of people involved in the project about Scrum framework. Their judgment is crucial since they are the figures that routinely deal with agile approach. To do this, they were asked to evaluate some aspects proper of the Scrum.

The first topic mentioned in the questionnaire is related to the perceived effectiveness of organizing the project into iterative cycles, i.e., sprints. Since sprints are one of the main peculiarities of the Scrum approach, it was considered essential to investigate their potency. As highlighted in Fig. 5.8, *strongly effective* is the most frequent assessment (45,5%), subsequently *effective* (36,4%) and *sufficiently effective* (18,2%). Then, the Scrum framework was evaluated in terms of Events. Thus, investigating on how worthwhile the Scrum ceremonies were deemed to be in order to achieve the sprint goal. Are Daily meeting, sprint Review and sprint Planning perceived as key moment by the team members or more as a waste of time? The answer of the questionnaire confirms the relevance of the Scrum Events and their efficiency. In fact, as highlighted in Fig. 5.9, 18,2% of respondents considered the Scrum ceremonies as *sufficiently* useful, 54,5% as *fairly* useful and 27,3% as *a lot* useful. Hence, it is possible to state that also Scrum Events are considered significant peculiarities of the project, even if in

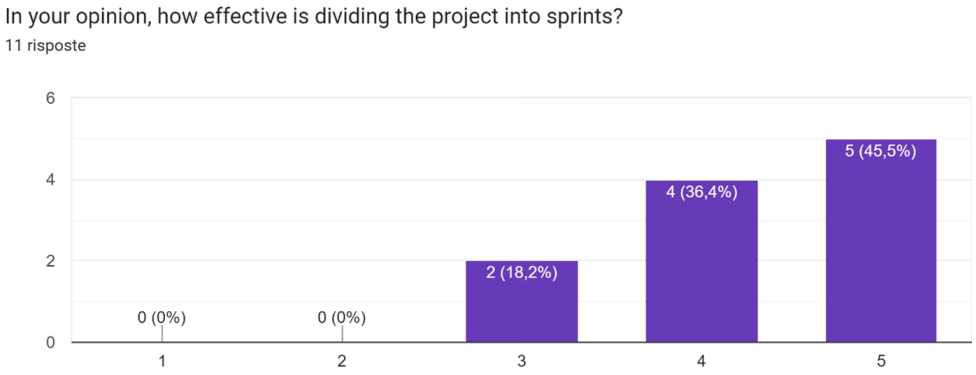


Figure 5.8: Sprint effectiveness.

this case the mode is on score 4 rather than score 5 as for sprints evaluation (Fig. 5.8). According with the interviewees, having a short time horizon to fill a goal stimulates the team to maintain the pace of work and focus on specific tasks reducing any waste of time. On the other hand, although Scrum recurrent meetings are still considered critical to reach the objective with success, it turned out that their efficacy is strictly related with their duration. Indeed, especially for developers, it is essential finding the right tradeoff between time devoted to organizing the work and the time dedicated to the actual development. Too long meetings are likely to affect the work of developers and slow down the process, with negative consequences on the sprint outcome.

In the case of study of this thesis, it emerges that the scheduling of the Scrum

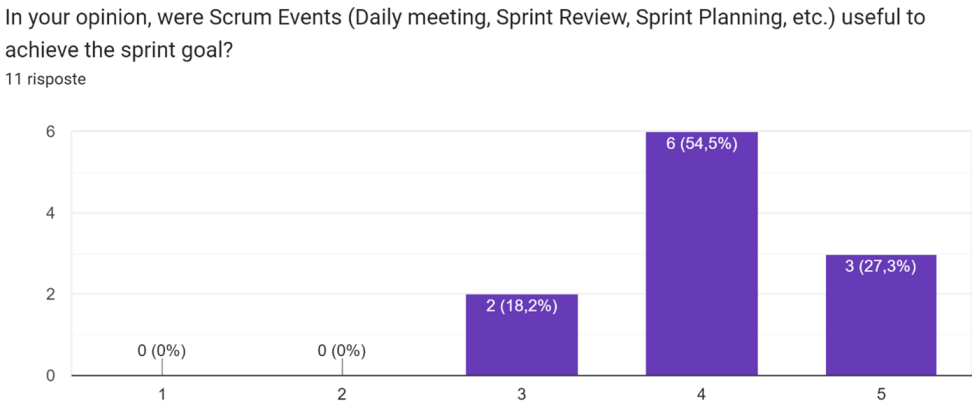


Figure 5.9: Scrum Events effectiveness.

Events was sometimes overwhelming. Fig. 5.10 illustrates the opinion of the team related to the amount of time left for actually working on the activities. It emerges that 18,2% of respondents figures the leftover time from meetings was *not much* to work on the activities and 45,5% states it was *sufficient*, whereas the rest assesses it as *fairly* and *a lot* adequate (18,2% of respondents each). It is important to point out that lower scores are assigned by developers.

Contrarily, the two-weeks duration is considered *effective* by most of the respon-

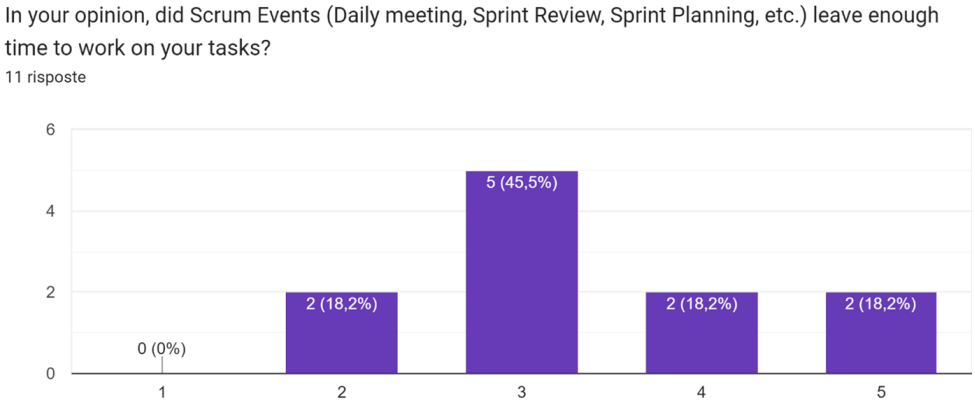


Figure 5.10: Duration of Scrum Events.

dents (90,9%), whereas just a person that thinks it is *too short*.
To sum up, the trend of considering managing the project within the Scrum as efficient continues.

After the sprint and the Scrum Events' efficacy, the questionnaire continued with the assessment of the Scrum roles. In particular, the importance attributed to the Scrum Master to let the IT project successfully proceeding has been studied. The result are reported in Fig. 5.11. Altogether it emerges that Scrum Master is considered as essential to achieve the project goal, with all respondents assigning the higher scores. They declared it as *fairly* important (63,6%) and the remaining (36,4%) as *a lot*.
Hence, strictly analyzing the perception the team has with regard to the Scrum framework, the result shows the Scrum Master as the more critical element of the framework, followed by the organization in sprints and then by the Scrum Events. However, all of Scrum peculiarities obtained high scores in terms of assessment in the project.

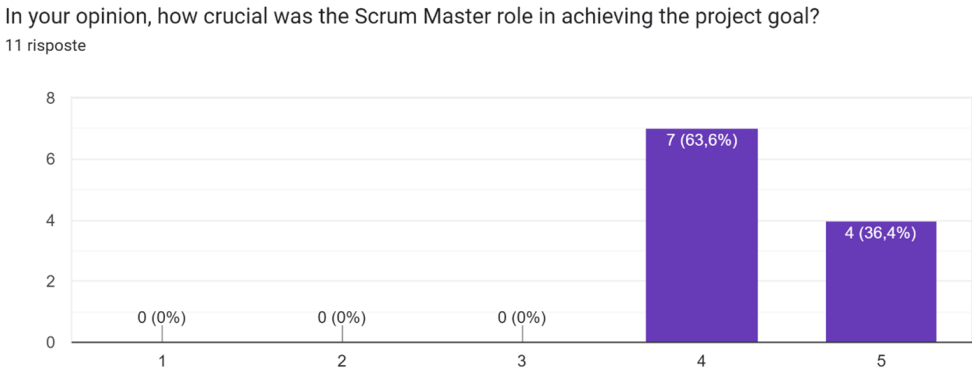


Figure 5.11: Scrum Master perceived importance.

Then, the study aimed to investigate which are the advantages and disadvantages of Scrum approach perceived by the team. The principal strengths of the Scrum are schemed in Fig. 5.12, according with the interviewed.

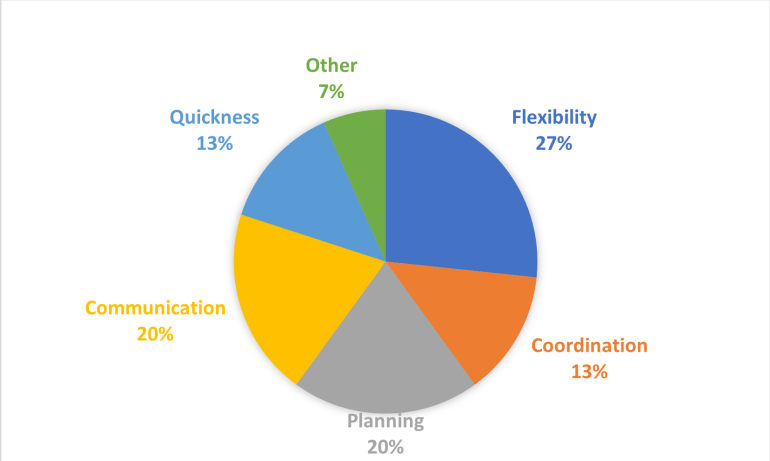


Figure 5.12: Strenghts of Scrum.

- Flexibility, i.e., the ability to change or be changed easily according to the situation, is the main gain the Scrum approach offers, according with 27% of respondents. This methodology of project management indeed provides handling the project in an iterative manner, consistently with agile mindset, therefore letting the stakeholders to ask for adjustments after the

project was already started. Owing to values as transparency, inspection adaptation, Scrum facilitates flexibility, allowing the team to adequately respond to changes in requirements without significantly impacting overall project progress. Indeed, the team demonstrated to be capable of successfully handling the introduction of new requirements, or any changes, while the product was already being developed. For example, in sprint 1 and 2 the workload clearly increased with the insertion of unplanned tasks (Fig. 5.4) but, thanks to the versatility of the project organization, the team realized them without significant process blocks.

- Coordination is the key point of any success project, and it turns to be essential also for project managed at very low level as those led by Scrum approach. A correct implementation of the Scrum framework eases the coordination mechanisms for synchronizing the project team, structuring their relations, and handling the activities. Therefore, contributing to the success of the project. Here comes back to be fundamental the role of the Scrum Master since this figure is accountable for the alignment of the team. The coordination should be understood both a team level and at the level of cooperation between individual team members.

- The continuous and constant planning of the activities to be carried out in the sprint is a crucial point of the Scrum. The respondents depict the possibility to have scheduled tasks and effort has a huge advantage, supporting the agile principle of avoiding waste of time. In fact, the tasks are clearly stated during the Sprint Planning and this facilitate the team to keep up. Planning in Scrum is collaborative, letting autonomy and self-management to developers. In turn, it increases efficiency. Although planning is important, being able to respond to changes is more, according with Agile Manifesto.

- A strong communication among the members is a critical aspect that enhance collaboration and cooperation. Co-working towards the same goal, the team member should share immediately any problems that may block the activity with the team, in order to immediately solve them. Among the respondents, 20% believes that the framework implemented in handling a project with Scrum eases this aspect. In fact, frequent meetings, and constant monitoring by the Scrum Master of the progress of the project, help the team to have a good level of dialogue. Basing on this study, the

Scrum approach has also the benefit of reducing the bottlenecks in the information flow. In particular, 18,2% of the respondents perceives Scrum *sufficiently* adapt to decrease bottlenecks, 45,5% *fairly* adapt and 36,4% *strongly* adapt, as depicted in Fig. 5.13.

- Quickness is another advantage that emerges in the study. The concept relies on the ability to advance with the project fast. According with 13%, of the respondents, Scrum permits to speedy develop, since any roadblock is fast solved. In turn, it significantly decreases bottlenecks and allows the development of the product smoothly. A consequent benefit is the completion of the project in a shorter time. This is allowed also by the reduction of wasting time as provided in the agile mindset.

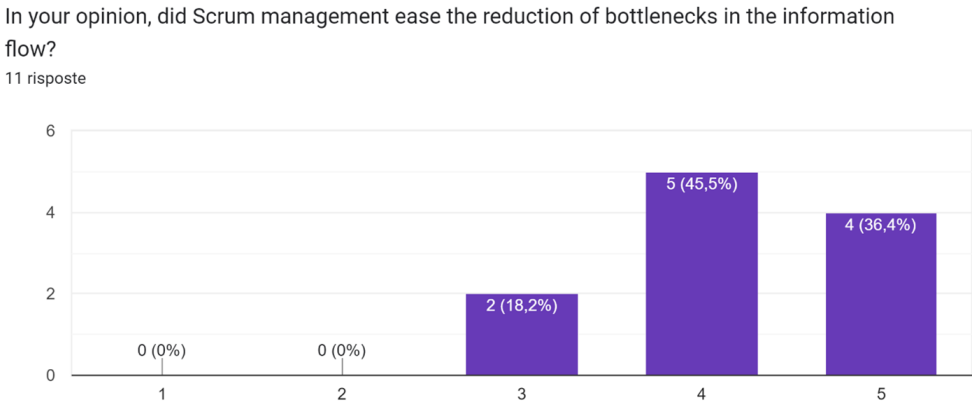


Figure 5.13: Reduction of bottlenecks.

With regard to the disadvantages, the outcome of the questionnaire is reported in Fig. 5.14. Up to eleven respondents, two declares to have *never* encountered difficulties in the organization of the project (18,2%), eight to have meet them *rarely* (72,2%) and just one *often* (9,1%). The main trouble with Scrum approach the team declared to have is in the duration and frequency of the meeting. Indeed, developers pointed out the subtraction of time in the development activities due to meeting too long. Again, this confirms the importance of keeping the meeting on time and avoid those useful, as already discussed before. Another aspect is the overlapping with other meetings when a member works simultaneously in more projects. Finally, the satisfaction of the team on the way the project has been managed is investigated. As results in Fig. 5.15, 81,8% is

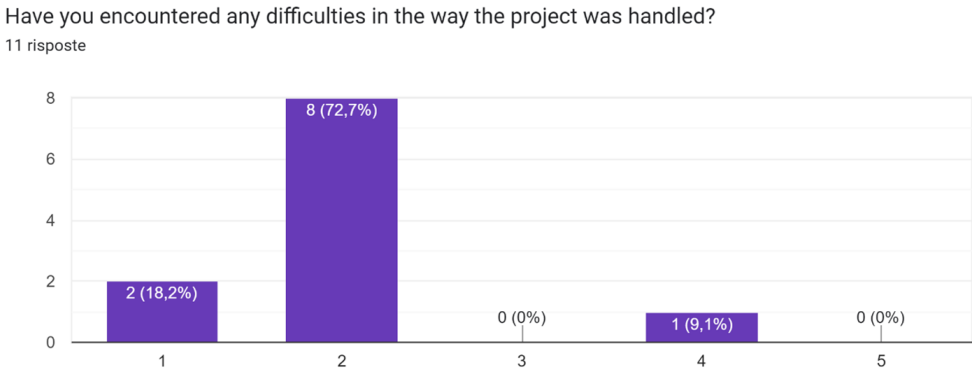


Figure 5.14: Difficulties encountered within the project.

fairly satisfied whereas 18,2% is *fully* satisfied. Furthermore, 63,6% declares that there is not another project management approach more suitable for the project, whereas 9,1% states that there is. The rest declares to not know the answer.

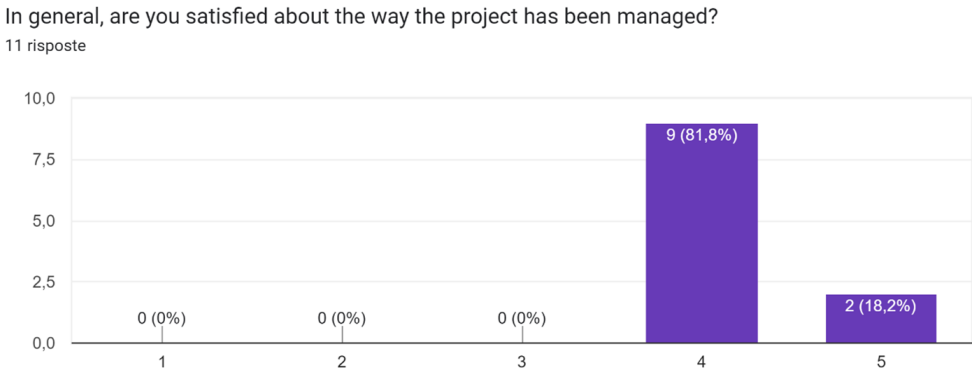


Figure 5.15: Team satisfaction.

5.3 Summary

The outcome of the analysis states the project as efficient both from a technical and managerial perspective. Concerning the first result, the efficiency has proved by the successful delivery of the project and the accomplishment of all developments on time. The team confirmed to be productive throughout the entire project, despite some sprints were more well-performed than others. With regard the process organization, the perception of the team about the Scrum approach

is positive as well. A correct Scrum implementation was essential for the team to react to unexpected events, changes, and new requirements without blocking the progress. It also emerges that the information flow efficiently streamed, reducing the possibility for blocking point to arise. Furthermore, an efficient communication within the team triggered the ability to fast solve roadblocks, because the chain provided by the Scrum facilitates the resolution. In turn, it also allows developers to quickly complete the tasks, reducing the time. The latter concept is also enabled by a proper and precise plan, as foreseen by Scrum framework.
The study points out some aspects about the management of the project that should be taken into consideration by organizations. A takeaway regards the duration of the meetings, since too short meetings run the risk to be inefficient. On the other way around, an excessive amount of time spent in reunions discourages development activities. Thus, it is essential to set the agenda rationally. Another point to carefully weigh is the duration of the sprint, given that it is the first step for handling the project in an efficient way.
Furthermore, the workflow of the activities must be reasonable, understandable, and as simple as possible. Avoiding cumbersome flows enhance clearness, thus encouraging agile value of transparency.
Constantly monitoring the activities is fundamental to control the project trend. The support of Jira graphs is here helpful and should be exploited by the Scrum Master and Product Owner. However, the accuracy and consistency of the charts may be weak owing to the workflow setting and thus they always need to be further analyzed and contextualized, for example weaving outcome from different charts.

Conclusion

Final Considerations

This thesis aims to analyze the effectiveness of the Scrum approach in managing an IT project and to identify which are the benefits of Scrum perceived by the team. Thus, a real case was studied, concerning an integration project of the spare parts section for the D2C e-commerce of a brand operating in domestic appliances industry. From the case study results and analysis, it emerges that Scrum management practices, together with an agile mindset, help the organization to successfully complete the project on time and on scope, without losing on quality. As proven by the results reached during the sprints, it is possible to state the project as well-performed. Indeed, the project was technically ready to go live at the planned date, satisfying all the requirements agreed with stakeholders. Furthermore, the project is marked as efficient in terms of easiness of keeping up with the activities, tasks clarity, sense of involvement in the team and in the project and efficacy of communication. Through the analysis of a questionnaire delivered to the team, it also emerged a positive assessment of Scrum effectiveness in handling the IT project. In particular, Scrum results thoroughly effective in every aspect of the framework. Dividing the project into sprints is considered efficient by most of respondents, since frequent deliveries help them to focus on the work and reducing any waste of time. Consistently with agile, the adoption of an incremental approach and iterative cycles allows the team to better adapt to changes, generating higher motivation to the project team and stakeholders' satisfaction [18]. Another important proved benefit of the Scrum framework is the systematic realization of the ceremonies. This aspect confirms that recurrent meetings facilitate communication and collaboration among the team, offering lightness to the project and waste reduction. Furthermore, the Scrum Events, especially the Sprint Reviews, enable the exchange of feedbacks between stakeholders and the team, identifying any non-conformities and modi-

fications, and permit a prompt intervention. It increases better control over the project, mainly in its scope. The presence of a coordination role as the Scrum Master also emerged as essential.
This work also points out what advantages and disadvantaged emerged in adopting Scrum approach. The main strengths of Scrum can be summarized in flexibility to adapt the work, coordination between team members, planning of the activities, quickness in advancing with the project and constant communication. Also, the study states that Scrum approach is useful to reduce the bottlenecks in the information flow. However, Scrum effectiveness is achieved only paying attention to some precautions, otherwise it runs the risk to hinder the process. As a matter of facts, meetings should be scheduled to last an adequate amount of time that enables the team to update and efficiently communicate without bothering the development activities. Still, monitoring the activities is fundamental to control the project trend and the support of charts is also vital. However, the accuracy and consistency of the charts should be check before drawing conclusions. Finally, this study contributes evidence about the relevance of how a project is managed, showing the importance of establishing an efficient organizational process.

Limitations of the Study and Future Research

It is noteworthy that this thesis consists of a single exploratory case study of a specific IT project in domestic appliance field, and therefore results cannot be generalized. Indeed, it is hard to state that Scrum approach is always effective, as emerged for this project. This work aims to contribute to show the importance and efficacy of having an effective management of the project, in particular through Scrum approach. Nevertheless, the findings may offer general learning, as well as important practical information and experiences to companies interested in applying the Scrum management for their projects. Furthermore, the research focuses on qualitative aspects of Scrum project management, without considering the budgetary dimension. This analysis was not carried out due to lack permissions for access to economic data. In addition, this study is limited by the fact that Scrum was not purely applied since some peculiarities in the way the project was handled were customized and adapted to the project.
To expand the research, it may be interesting to compare this study with similar projects handled within traditional or other agile methodologies and investigat-

ing how Scrum approach performs in case of more complex IT project. Another suggested future scope of research is to apply the study to multiple projects, thus spreading the questionnaire to a wider sample in order to validate the findings. Furthermore, assessing the effectiveness of the project also from a budgetary perspective may give a broader view of the study. Finally, it may be interesting to analyze how much the effectiveness of Scrum obtained in the project is related with the fact that it is not pure.

References

[1] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK®Guide) – Fifth Edition*, 2013.

[2] Atkinson R., *Project Management: Cost, Time and Quality, Two Best Guesses and a Phenomenon, its Time to Accept Other Success Criteria*, 1999.

[3] Baratta, A. *The Triple Constraint: a Triple Illusion*, 2006.

[4] Badewi A., *The Impact of Project Management (PM) and Benefits Management (BM) Practices on Project Success: Towards Developing a Project Benefits Governance Framework*, 2015.

[5] Thesing T., Feldmann C., Burchardt M., *Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project*, 2020.

[6] Mészàros A., Csiszàrik-Kocsir A., *The Difficulties of Implementing Agile and within it the Scrum Framework*, 2022.

[7] Lalic D. C., Lalic B., Delic M., Gracanin D., Stefanovic D., *How Project Management Approach Impact Project Success? From Traditional to Agile*, 2022.

[8] *Manifesto for Agile Software Development*, 2001. https://agilemanifesto.org/.

[9] Gemino A., Horner Reich B., Serrador P. M., *Agile, Traditional, and Hybrid Approaches to Project Success: Is Hybrid a Poor Second Choice?*, 2021.

[10] Wysocki R. K., *Effective Project Management: Traditional, Agile, Extreme, Seventh Edition*, 2014.

[11] Anderson D. J., *Kanban: Successful Evolutionary Change for Your Technology Business*, 2010.

[12] Scaled Agile, *Achieving Business Agility with SAFe®5 A Scaled Agile, Inc. White Paper*, 2021

[13] Rising L., Janoff N. S., *The Scrum Software Development Process for Small Teams*, 2000.

[14] Abrahamsson P., Baskerville R., Conboy K., Fitzgerald B., Morgan L., Wang X., *Agile Processes in Software Engineering and Extreme Programming*, 2008.

[15] Schwaber K., Sutherland J., *The Scrum Guide*, 2020.

[16] Matthies C., Dobrigkeit F., *Experience vs Data: A case for More Data-informed Retrospective Activities*, 2021.

[17] Puliti G., *Guida Galattica per Agilisti Parte III: il Framework Scrum*, 2016.

[18] Azanha A., Argoud A. R. T. T., Camargo Junior J. B., Antoniolli P.D., *Agile Project Management with Scrum - A Case Study of a Brazilian Pharmaceutical Company IT Project*, 2016.