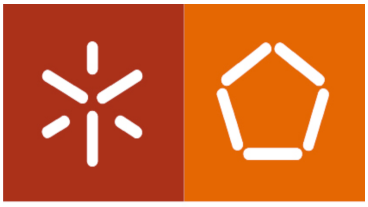

Universidade do Minho
Escola de Engenharia

Maria José Barros Araújo

# Improvement of performance indicators after Scrum re-implementation in software development teams

Master Thesis

Master's in Engineering Project Management

Work done under the guidance of:

**Professor Anabela Tereso**

**Professor Pedro Ribeiro**

January 2019

# ACKNOWLEDGEMENTS

I would first like to thank my thesis advisors, Anabela Tereso and Pedro Ribeiro, of the Engineer School at the University of Minho, who supported me throughout my path and always answered to all of my doubts.

I would also like to thank my mentor, Timothy Bishop, who held my hand through my learning path, watched me take little steps and then gave me the opportunity to walk alone, never leaving my side. To Eleanor Wilson, the person that grew worked and fought for Scrum, alongside me.

I would also like to thank my Team members, who believed I was capable to be a good Scrum Master, always supporting and listening to me. Without you guys, this project would never have had existed.

To Martina, for all the patience, hard work, and for being by my side, through all my doubts, fears and insecurities. To all my Erasmus+ friends - Laura, Jeroen, and Martón - thanks for making me laugh, and for always encouraging me to never give up.

Finally, I must express my gratitude to my parents and to Beatriz and Helena for supporting me from far and for always giving me the courage to keep going on, even though not understanding anything about my research or about my job, they always kept me going and made me feel loved.

This accomplishment would not have been possible without them. Thank you.

# ABSTRACT

The business world is more competitive than ever, and companies must adapt to a very fast passed consumer behaviour. Small and medium companies must be quick on their feet and have the ability to rapidly answer to the shifts in trends and products.

In order to do so, businesses have shifted from Traditional project management to an Agile project management. Even though Agile project management has become very trendy, implementing an Agile framework requires work and an adaption period.

Living Map adopted an Agile framework – Scrum – one year ago, but because they changed their Scrum Master, the process gradually broke and became problematic.

This research was proposed by the company, to re-implement Scrum and analyse the changes on the performance indicators of the Software Development teams. The main goal was to collect data after each Sprint completion and compare it to old data collected previously by the researcher whilst being an intern.

The first step was to re-define and re-implement Scrum within the two development teams. The researcher acted as the Scrum Master of both teams in order to be closer to them.

This dissertation describes the re-implementation of Scrum, the adaptive measurements that were made during a 5-iteration cycle and the changes on the performance indicators that originated from it.

In general, not only did every performance indicators had a significant growth but also, the understanding of Scrum increased, and the overall satisfaction of the Teams also improved.

**KEYWORDS**: Agile; Scrum; Performance; Research and Development.

# RESUMO

O mundo de negócios tem-se tornado mais competitivo do que nunca. As empresas têm de se adaptar às alterações rápidas do comportamento dos consumidores. Pequenas e médias empresas têm de ter a habilidade de responder rapidamente às mudanças das tendências e dos produtos.

De forma a conseguirem fazer isto, há uma mudança da gestão de projetos tradicional para gestão de projetos ágil. Porém, implementar uma mentalidade ágil requer esforço e um período de adaptação.

A empresa *Living Map* adotou um *framework* ágil - *Scrum* - há um ano atrás, mas por terem mudado de *Scrum Master*, o processo acabou por se deteriorar e tornar problemático.

A pesquisa de reimplementar o *Scrum* em duas equipas de desenvolvimento de *software* foi proposta pela empresa, e o investigador desempenhou o papel de *Scrum Master* de ambas as equipas. O objetivo principal foi recolher dados após o fim de cada *Sprint* e comparar os mesmos com dados recolhidos de *Sprints* anteriores.

O primeiro passo foi redefinir e reimplementar *Scrum* em duas equipas de desenvolvimento de *software*. Esta dissertação descreve a reimplementação de *Scrum*, as medidas adaptativas que foram tomadas durante um ciclo de 5 iterações e as mudanças dos indicadores de performance que ocorreram.

Em geral, não só os indicadores de performance tiveram um crescimento significativo, mas o conhecimento sobre *Scrum* e a satisfação das equipas melhorou.

**PALAVRAS CHAVE**: Agile; *Scrum*; Performance; Investigação e Desenvolvimento.

# INDEX

# FIGURE INDEX

# TABLE INDEX

# SYMBOLS AND ACRONYMS

| Symbol | Description |
|---|---|
| CTO | Chief Technology Officer |
| COO | Chief Operation Officer |
| DEV | Development |
| GDPR | General Data Protection Regulation |
| GIS | Geographic Information System |
| GPS | Geographic Positioning System |
| IoT | Internet of Things |
| IPS | Information Position System |
| LM | Living Map |
| OOH | Out of Home |
| PO | Product Owner |
| ROI | Return on Investment |
| SAAS | Software As A Service |
| SDK | Software Development Kit |
| SM | Scrum Master |
| TA | Technical Architect |
| UCD | User Centred Design |
| WIP | Work In Progress |

# 1. INTRODUCTION

The introduction is structured in five parts. Firstly, a brief background of the study will be presented, followed by the company's description. In part three the objectives and motivation for the research are presented, and in part four there is an explanation of the research methodology adopted. Finally, in part five the structure of the document is shown.

## 1.1 Background

The business world operates in a global, chaotic and rapidly changing environment. In order to succeed in such a competitive market, companies need to be able to quickly respond to new opportunities, changing situations and emerging new products and new services (Sommerville et al., 2011).

It is crucial for organizations to practice better project management. Organizations should have a good knowledge about project management practices, and which ones are best fit to their sector of activity. A study about project management practices in private organizations refers that the 10 most used practices for the information and communication sector are: kick-off meeting; progress meetings; Gantt chart; activity list; baseline plan; progress report; change request; client acceptance form; project scope statement and requirements analysis (Tereso, Ribeiro, Fernandes, Loureiro, & Ferreira, 2018).

To embrace this hectic environment, software development practitioners attempt to keep up to speed to their surrounding with processes that not only respond to change but also embrace it as a market advantage (Cohen, Lindvall, & Costa, 2003).

Software is a component of basically everything we use. The high demand for newer and faster products makes the software development business very competitive. To be able to compete in this volatile market, rapid development and delivery are two crucial must-haves. Besides being fast, the business must not put aside the quality of the product.

Agile methods are born as a solution for software development needs by being incremental development methods. In Agile, customer releases are made in short iterations, with small increments. The customer is involved through the process in order to get constant and rapid feedback on requirements. Documentation is minimized by the use of informal communication (Sommerville et al., 2011).

Between 1980 and 1990 there was a common belief that the best way to achieve better software was through careful planning, documenting, inspecting and testing. This view came from developing software systems that were very big and had a long-life period. Such methods, with heavyweight, plan-driven development approach made sense because of all the variables in the game. Teams that were very big, dispersed geographically and most times were not allocated in the same company, but still, had to be coordinated.

Traditional methods began with a complete and exhaustive analysis of user requirements. After a long time of interaction with users, engineers would create a detailed list of features and requirements. This information had to be very well documented, so engineers could create the optimal architecture for the system, and then finally implement the design, test it and ship it (Beck, 1999).

However, these traditional methods did not apply to small and medium-sized business and were creating a general dissatisfaction within the business. The industry and technology moved too fast, requirements were changing at rates that did not allow traditional methods to keep up and clients did not know what they wanted in advance, having only a general idea of what their objective was.

Due to the market uncertainty, the use of traditional methods was frustrating to developers. A usual case was that developers would spend a year planning, documenting and studying the market and suddenly would have to change, based on new requirements or new product's launching. The Waterfall model was supposed to fix this problem by freezing requirements and not allowing any change. By doing this, the project completion could be a success, but the project scope and the client satisfaction could be a failure. This happened because there was no time to deal with feedback or a change in the requirements. Based on these needs, the Agile Manifesto was created in 2001. From there, a variety of Agile Methodologies was created.

## 1.2 Living Map

Living Map was founded in 2010 by Tim Fendley, as a sister company of Applied Wayfinding, that focuses in urban design, in order to serve as a research and development department. Right now it is a digital mapping and data platform that provides visual solutions to companies.

Living Map technology can deliver a range of geographic-based digital solutions working from indoor and outdoor mapping and navigation, delivered on digital formats - signs, screens, websites, and mobiles or embedded into existing systems or apps.

It’s a small company, based in the United Kingdom, with a team composed of thirty-four specialists, experts across environmental graphic design, cartography, typography, product design, urban planning, project management, and implementation.
In order to better deliver projects and respond to customer needs, Living Map has implemented Scrum to their development teams.

## 1.3 Objectives and Motivation for the Research

Being Agile is a very trendy business word. Even though an increasing number of companies claim to be Agile, not many understand what that concept implies.
Living Map adopted Scrum, one of the most know agile frameworks, not only for their Software Development Team but also for their Geographic Information System (GIS) Team and for their Design Team. When the researcher joined Living Map as a business analyst, to help the current Scrum Master organize work, gather requirements and sort dependencies between teams, the teams had been doing Scrum for nine months and were still learning how to integrate the process with their reality.
After a month of joining the company, the Scrum Master at the time was promoted to Chief Technology Officer (CTO), and due to budget constraints, the new Scrum Master (SM) appointed was also the Technical Architect (TA).
After this change in job roles, some Scrum practices were still being followed but many others were being overlooked, which broke the process and created general discontent on the team and management level, leading to the leave of the Scrum Master/Technical Architect.
Together with the Management Team and the Development Team, it was proposed that the researcher would assume the role of Scrum Master, re-implement the process and evaluate the benefits that came with it.

## 1.4 Research Methodology

The methodology adopted for this dissertation was Action Research. This type of approach is known for being participative, taken by individuals that have the same purpose and context specific. The main goal of action research is to find ways to improve how something works, creating knowledge (Serie, 2010).
In order to know if a suggestion is actually working, Action Research can be used. This fits with the Agile view. If you need to test something, and you might fail, it's better to fail fast and fail with a full understanding.

There are five phases to Action Research - identify a question, action planning, implement a solution, determine if it works, and specify learnings - and this aligns with the Scrum and its Sprints.

In phase 1, the researcher states the aim and purpose of the research. This is done by adopting an outsider point of view. Later, the researcher will have a role in the research. Next, there is the action planning, phase 2, where the overall plan is drawn. These improvements or solutions can be created from scratch, but they can also be from existing ones.

In phase 3, the solutions planned in phase 2 are taken into action within a certain time frame, and the data originated in phase III is gathered in the next phase (phase 4), analyzed and compared to the initial data to understand the changes that occurred. In the last phase (phase 5), the new knowledge acquired is presented. An action research methodology can have various cycles where the phases are repeated.

For this research, the process was as it follows. In the beginning, the question was chosen: "Will Scrum re-implementation improve performance within the development Teams?". The second phase was to implement the right practices of the Scrum framework - this knowledge already existed, although it was adapted to fit the company and the team needs - during five iterations. In each iteration, feedback was taken from Retrospective, data was analysed, and when needed, actions were refined and taken into the next iteration.

After the five iterations, all data was gathered, and compared to the last three iterations prior to the solution was implemented. In the end, the conclusions were presented.

## 1.5 Structure

This document is structured in sixth parts. The first chapter is the introduction to the theme, where topics like the objectives, motivation and the research methodology are mentioned.

In the second chapter, there is the Literature Review, where the knowledge about traditional methodologies and the agile methodologies is shown. In this chapter the focus is Scrum, an Agile framework.

In the third chapter, the case studied (Living Map) is presented including a SWOT and PEST analysis, a competitor’s analysis, the organisational structure, the working areas, the process of creating a map and the JIRA software used.

The next chapter (fourth) identifies the problem that this research aims to improve, and the solutions implementation are presented.

The fifth chapter gathers all the data collected from the results, and on the last chapter a conclusion is drawn.

# 2. LITERATURE REVIEW

This chapter's aim is to show the knowledge about the theme and to report on the existing literature on the theme.

This chapter is divided into two different types of methodologies of project management. Firstly, there will be a focus in traditional methodologies and some of the most relevant frameworks – Waterfall and the Spiral model.

The second part of this chapter focuses on Agile methodologies, and the relevant models – Extreme Programming, Kanban and Lean Development, and finally Scrum.

Since the research talks about the implementation of Scrum, this framework has a bigger emphasis in the chapter, focusing on topics like – Scrum Roles, Scrum Ceremonies, Scrum Artifacts and estimation.

## 2.1 Traditional Methodologies

In traditional methodologies, there are several steps one must take. The process starts with the documentation of requirements, studying and analysing a solution, developing, and testing. The requirements have to be well defined and fixed at the beginning of the development process, making the process frustrating to many developers due to the highly changing market that software development fits in (Awad, 2005).

Traditional methods create a disciplined process in order to make the software development process predictable and efficient, having the following characteristics:

- Predictive approach, where the work is first planned in high detail. A large amount of time is spent on drawing the system, understanding the requirements and coming up with a solution that fixes it efficiently. These drawings specify how the product should be built, the schedule it should follow and the resources that should be used;
- Comprehensive documentation, where all the client requirements are collected before the coding starts;
- Process oriented, since the goal of the traditional methodologies is to define a process that can be repeated and that works well;
- Tool oriented.

There are a few traditional methodologies such as the Spiral model, the V model and the Waterfall model.

### 2.1.1 Waterfall

The Waterfall model is a sequential software development process that involves five phases - analysis, design, implementation, testing, and maintenance. The progress is seen flowing downwards like a waterfall through all the listed phases. The phases must be executed by that order to achieve success in building software, meaning that one phase is dependent on the completion of the phase preceding it (Awad, 2005).

The stages description is as follows, and shown in Figure 1 (Bassil, 2012):

- Analysis, where a complete and comprehensive document describing how the software should behave is created. It has an analysis, both on the business and on the system, in order to fully identify the functional and non-functional requirements that the client expects. This document includes the objective of the project, the scope, the perspective, functions, attributes, user tasks, specifications, interface, and database requirements;
- Design, the software developers and designers work together to achieve the software solutions. Here are included tasks like algorithm design, software architecture, database schema, and user interface definition;
- Implementation, where the developer takes de requirement document and turns it into an executable product. This is when the code is written;
- Testing or verification and validation is the stage where the solution is checked to see if requirements and specifications were kept and that the solution meets its purpose;
- Maintenance, where the solution is modified after the delivery to refine the product, correct errors or improve general performance and quality.

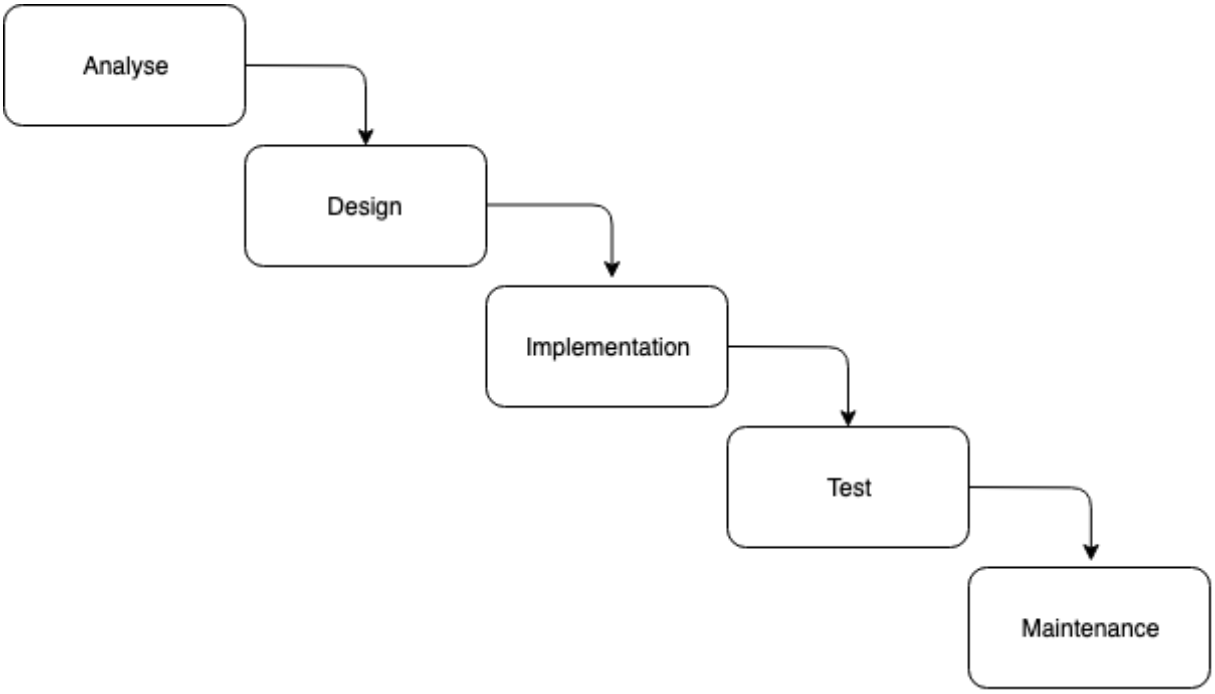


**Figure 1 - Waterfall Framework**
(Murugaiyan, 2012)

When using the Waterfall model, all the requirements are clear before the development phase starts. If the client wants to change a requirement, the change will not be implemented in the development phase. This may result in project success but in client dissatisfaction. Each phase is completed in a specified period of time, and only after that phase is completed, it moves to the next phase. This means that if a phase gets delayed, all the project gets pushed back. The Waterfall model is very easy to implement and it requires minimal resources to be implemented (Murugaiyan, 2012).

### 2.1.2 Spiral Model

The spiral model is a traditional methodology that like the waterfall model is composed of phases - Planning, Risk Analysis, Engineering, and Evaluation. The software project continually passes through this stages in iterations (the Spirals) (Awad, 2005).

- Planning is where the objectives are identified and agreed upon;
- Risk Analysis where the risks are identified, analysed and information for risk mitigation is gathered;
- Engineering, where the software is produced;
- Evaluation allows clients to evaluate the output of the project to date before the project continues to the next spiral.

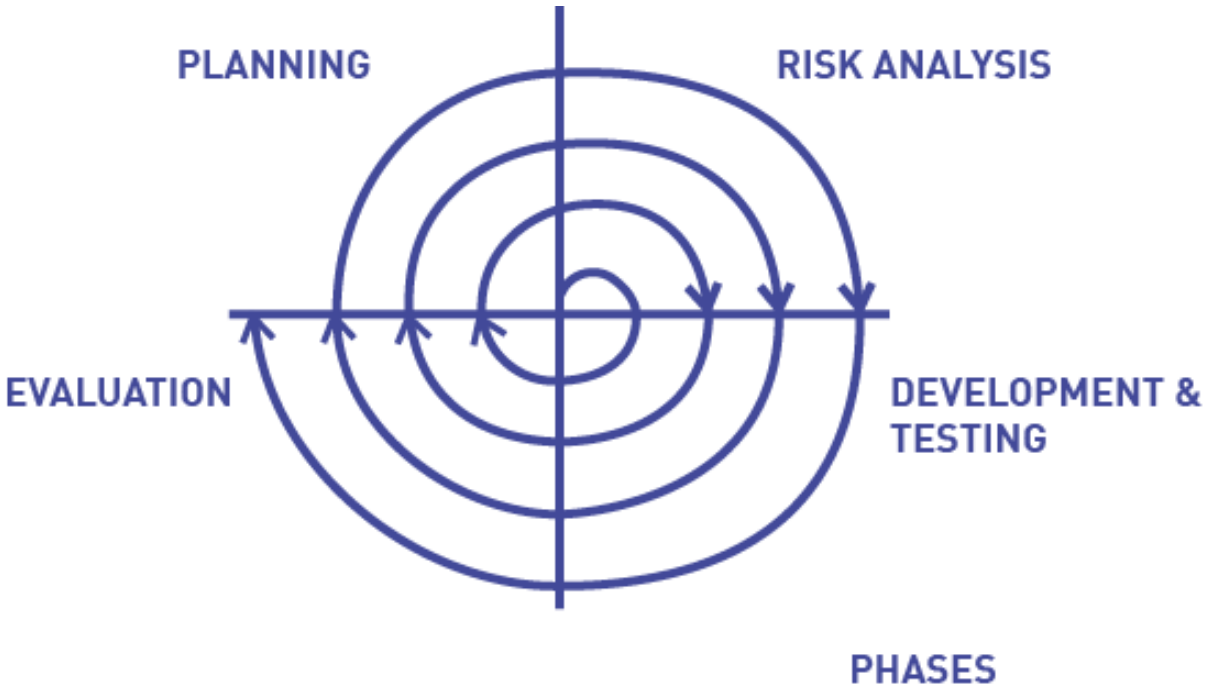


**Figure 2 - Spiral Model**
(Boehm, 1988)

The model shown in Figure 2, has the base concept that each cycle involves the same stages sequence, for each increment on the shippable product and for each of its level of production. Each spiral starts with the identification of the objectives of the increment that will be produced, the different ways to implement the product and the constraints imposed on the application of alternatives. Following this, there is the stage where the alternatives are going to be evaluated. Here, the areas that are uncertain and might cause a risk for the product development are identified. If these risks are identified, there will be an analysis of cost-effective strategies to mitigate the risks.

After identifying the risks, there will be phasing to implement and develop the software. Each level of software specification is followed by a test and validation phase.

Each cycle of the spiral is basically a review involving those interested in the product development. This review covers every product produced in previous spirals (Boehm, 1988).

This model has some advantages like how a high amount of time allocated in risk analysis and that the software is produced early in the cycle, but it is a costly model to use since it requires very specific expertise and where the success of the project is mostly dependant on the risk analysis phase. The Spiral model is a good model for large and mission-critical projects but does not fit well in smaller projects.

## 2.2 Agile Methodologies

The Agile movement officially started in 2001, when seventeen software developers got together and identified what needed to be changed in the software development process. Their intention was not to create a unique way to work, but to establish their common ground with a set of values and principles. By the end of this gathering, the Agile Manifesto was created. In the manifesto, the participants underlined four common values (Fowler & Highsmith, 2001).
They moved the focus from the process and the tools to the team relationships, work environment, and spirit. The individuals and the interactions overruled the processes and tools. The human role was the focus.
The main goal of the team was to deliver in a steady and frequent pace, software that worked. The code should be simple, clean and as advanced as possible in order to diminish the documentation and the paperwork. The working software should win over the comprehensive documentation.
The relation between the customer and the developers was more important than the contracts. It focused on delivering immediate business value as the project starts, reducing the risks of not fulfilling the contract.
The team that develops the software and the customer's representatives have to be well-informed and have the knowledge to consider and implement possible changes during the life-cycle of the project. The contracts are written in order to allow these enhancements. Responding to change has a greater focus than following a plan.
Described in the Agile Manifesto, there were 12 principles that developers should have in mind (Fowler & Highsmith, 2001):

- The highest priority is to satisfy the customer through early and continuous delivery of valuable software;
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage;
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
- Business people and developers must work together daily throughout the project;
- Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done;

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation;
- Working software is the primary measure of progress;
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely;
- Continuous attention to technical excellence and good design enhances agility;
- Simplicity – the art of maximizing the amount of work not done – is essential;
- The best architectures, requirements, and designs emerge from self-organizing teams;
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

When compared to traditional methods, Agile brings a number of development techniques that add business value (Kelly, 2012):

- Enhanced responsiveness to the business and market needs – Agile methods are built around the premise that change in requirements should be embraced instead of ignored;
- Increased Return On Investment (ROI) because working software is deliverable and ready to be used earlier in the development cycle;
- Reduced risk – since Agile has frequent deliveries of software, the amount of feedback and interaction with the customer rises. By having more feedback, the Team can accommodate changes earlier in the development process, tackling the risk sooner with lower cost;
- Improved quality;
- Better project governance because the progress of projects is more transparent. By using the Agile "Toolkit", and having daily meetings, visible and accessible work, everyone understands what is being done and where the process is at.

From the Agile Manifesto a variety of Agile Methodologies were born such as Scrum, Extreme Programming, Lean Development Software and Kanban that will be discussed next.

### 2.2.1 Extreme Programming

Extreme programming was introduced by Beck in 1998 and is one of the most known and used Agile methods. It was named as it was because the approach pushes recognized good practices to an extreme level.

There are twelve rules on Extreme Programming and all of them are really precise (Beck, 1999).

- The planning game – the customers decide the scope and the timing of the releases. This is based on estimates provided by the developers;
- Small releases – the first version of the system is put into production only after a few iterations and new releases are made often;
- Metaphor – both customers and programmers decide on the shape of the system;
- Simple design – only the necessary design is done so the system can communicate what the programmers intend to communicate with no duplicate code;
- Tests – developers write the acceptance tests before writing the code and customers write functional tests. At the end of each iteration, all of them should run;
- Refactoring – every time a developer finds an improvement that should be made into the code, he should do it immediately. This keeps the code simple and maintainable;
- Pair programming – every code is written by two people at one screen;
- Continuous integration – the new code is deployed to the current system after a few hours;
- Collective ownership – every developer can improve any code anywhere if they see the opportunity because the code is owned by the whole team;
- On-site customer – a customer works with the team during the development process. He is on site to answer questions and to perform acceptance tests;
- 40-hour weeks – developers should not have to do overtime;
- Open workspace – the workspace is large, and the developers work in small cubicles.

It is said that the strength of XP does not come from the twelve practices by itself, but all of them combined (Cohen et al., 2003). XP embraces agile practices with the following practices (Sommerville et al., 2011):

- Incremental development – there are small and frequent releases. The requirements are based on simple customer stories that are the base of the decision on what functionality should be included in a system shippable increment;
- Customer involvement – since the client is present during all the development process;
- People over process – are supported by pair programming, collective ownership and only work 40-hours;
- Change – is embraced by having small and regular releases to the customer, test-first development, refactoring and continuous integration;
- Simplicity – is supported by regular refactoring and using simple designs.

### 2.2.2 Kanban

Kanban originated in the late 1940s when Toyota started optimizing its engineering processes based on the model supermarkets were using. Kanban is the subsystem of Toyota Production System, which was created for inventory control.

Kanban levels inventory with actual consumption, just like in a supermarket. A consumer goes to the supermarket and only buys the exact quantity needed at the required time.

The Kanban system worked following these steps (Sugimori, Kusunoki, Cho, & Uchikawa, 1977):

- A form order called Kanban is used. The form order can be a conveyance Kanban, when going from one process to the next process, or a production Kanban, if it is used to order production of the portion used by the process. All the Kanban cards are attached to the containers;
- When the content of a container starts being used, conveyance Kanban is removed from the container and taken to a stock point to pick up this part. The conveyance Kanban is attached to the container holding this part;
- The production Kanban in the container is removed and become a piece of dispatching information.

The Kanban system was implemented instead of other computerised system in order to:

- Reduce cost of processing information;
- Acquire facts rapidly and precisely;

- Limit surplus capacity.

Kanban gradually eliminates the usage of iterations and sprints by continuous delivering high priority features instead of planning work to fit into a rigid time-boxed iteration.

Agile teams are able to leverage the just-in-time production by matching the amount of work in progress (WIP) to the team's capacity. By doing so, the teams have more flexible planning options and more transparent through the development cycle.

The teams work with a board - used to visualize work and optimize the flow of work within the team - in order to maximize collaboration and accessibility from multiple locations. The board main goal is to ensure the team's work is visible, their workflow is standardized and that all blockers are immediately resolved. A basic board has three columns (Figure 3) - To Do, In Progress and Done. Work is represented with a card on the Kanban board to allow team members to trace the card's workflow in a visual manner.

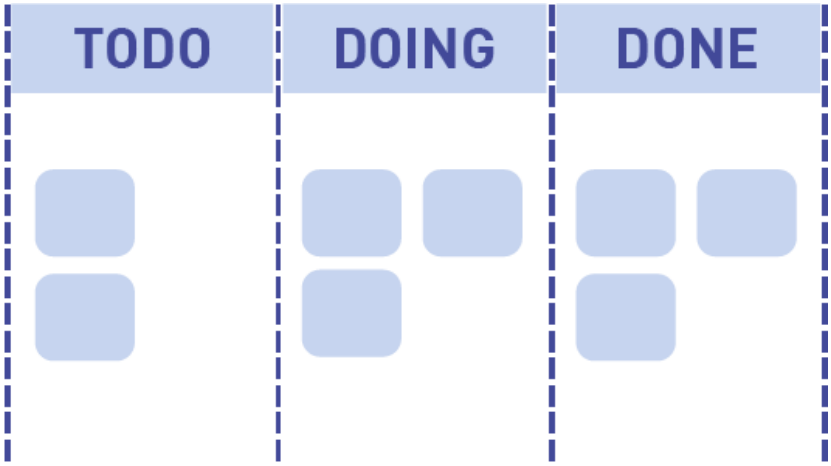


**Figure 3 - Kanban Board**
(COHN, 2005)

There are five core practices to Kanban implementation (Raju & Krishnegowda, 2013):

- Visualise the workflow - the Kanban screen allows teams to visualize the workflow of the work increasing transparency. By using the screen, team members can see what others are working on and offer help before starting a new task. It's also an easier way to let teams organize themselves, by identifying bottlenecks and take steps to resolve the flow;
- Limit work in progress - Kanban does not limit the time that a task might take but enforces a limit on WIP;
- Measure and manage flow - the cycle time is the time from when the work begins until it is delivered. It's a way of measuring the process capability;

- Make all process policies explicit - within the teams, they are certain policies on how a work should be done. These policies should be explicit in order to control the flow of the work and help team members establish their knowledge on what needs to be done;
- Improve continuously - Kanban encourages improvement on the process.

### 2.2.3 Lean Development

Bob Charette creates Lean Development after the success of Lean Manufacturing in 1980, with the belief that in order to be truly agile, a company must change the way it works from the top down (Cohen et al., 2003).

Lean Development has twelve principles that mainly focus on management strategies (Highsmith, 2002):

- The company highest priority is satisfying the customer. The Team must have practices defined to determine customer priority and others to listen to their responses. If customer expectation is not met, then the project is a failure;
- Always provide the best value for the money. Software should be a solution to a problem or a new opportunity at a reasonable cost. The main goal is value and not perfection;
- Customer must actively participate in the process in order for successful development;
- Every project is a team effort where the key to innovative, fast cycle deployment is multi-disciplinary teams rather than isolated individuals;
- Everything can change;
- Domain, not point, solutions – one-of-a-kind system is too expensive and cannot be used for other situations. In order to increase usability, the system should be applicable across multiple domains;
- Complete, do not construct;
- An 80 percent solution today instead of 100 percent solution tomorrow because markets change so rapidly;
- Minimalism is essential – waste should be eliminated by minimizing documentation, having small teams co-located and keeping the product scope focused;

- Needs determine technology;
- Product growth is feature growth, not size growth;
- Never push Lean Development beyond its limits.

### 2.2.4 Scrum

Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value (Schwaber & Sutherland, 2017).

This term was borrowed from Rugby. A Scrum happens when players for both teams huddle close together in an attempt to advance down the playing field. The term was used to explain the benefits of self-organizing teams in product development.

Scrum is one of the more widely used Agile Methods, alongside XP. In 1996, Ken Schwaber said that Scrum accepts that the development process is unpredictable. After inspecting the process, scientists concluded that rather than being repeatable, defined and predictable, most of the processes were unpredictable and unrepeatable. From this, the difference between a defined process - one that is predictable due to the fact that performs the same every time - and an empirical process - one that is chaotic and in constant need of measurement and control. Schwaber's framework is iterative and incremental. By using Scrum, companies are allowing for an interaction with the environment, leading to changes in project scope, technology, cost, and schedule (Schwaber, 1996).

Scrum is based on empiricism – where it is assumed that knowledge comes from experience and decisions are made and based on what it is known. It has three steady pillars – transparency, inspection and adaption.

- Transparency – every piece of work has to be clearly defined and every part involved on the project has to know about it.
- Inspection – every piece of work must be inspected with the necessary frequency to ensure the best quality.
- Adaption – it represents the capacity to adapt the project to the business needs.

These three values come to life when all Scrum values are respected – commitment, courage, focus, openness and respect (Schwaber & Sutherland, 2017).

Scrum has the following key principles (Schwaber, 1996):

- Small teams that maximize communication, enable knowledge sharing and minimize overhead;
- Adaptability to the environment, in order to ensure that the best product is produced;
- Frequent builds and releases, that can be inspected, adjusted, tested and documented;
- Breaking work into small pieces;
- Constant testing and documentation of a product at the same time as it is built;
- Ability to declare a product "done" whenever it is required.

Scrum assumes that the software development environment is too complex and unpredictable to be planned with a great deal of detail in advance. The solution is to apply empirical process control to ensure that the three pillars mentioned above are steady.
When talking about Scrum, one must mention it's roles, ceremonies, artifacts, and rules. The next sections will cover all of them.

### Scrum Roles

There are three Scrum roles - Product Owner, the Team and the Scrum Master. Scrum teams have and say and choose how to best accomplish their work because they are the experts on the subject. Furthermore, every member of the team should have the knowledge required to accomplish the work without having to depend on other members.
These roles are the people committed to the project. Within the business, there are external and internal parts interested in the project, but Scrum makes a very strong distinction between them. The framework makes sure that the ones responsible for the success of a project have more authority to do what is required and those who are not responsible cannot interfere (Schwaber, 2001).

#### - *Product Owner*

The Product Owner (PO) is responsible for representing the interests of all the parts with a stake in the project as well as maximizing the value of the work. The PO gathers all the initial requirements from the clients and creates the Product Backlog, that will later be used to ensure that the most valuable functionality is prioritized and built first (Schwaber, 2009).
To correctly manage the Product Backlog, the PO has to:

- Order items to best achieve goals and missions;
- Optimize the value of the work the Team performs;

- Make sure that the product backlog is visible, transparent and clear;
- Ensure that the Team understands what is needed.

It is very important that the organization respects the PO decisions since he uses the backlog to give the highest priority to requirements that are of highest value to the business, and constantly adjusts the product in response to changing business conditions.
The Product Owner voices what the customer asks for – his responsibility is to communicate the vision to the team, and this is why he needs to work with the team on a daily basis.

- *Scrum Master*

The Scrum Master is the protector of the team and the process. He is the primary interface between the Product Owner and the Team. He has to make sure every member of the Team understands the framework and respects the agile values.
One of the most important responsibilities of the Scrum Master is making sure the development team can work without distractions, that can be internal (e.g. the Chief Executive Officer wanting a new piece of work done) or external (e.g. noisy background).

- *Team*

The Team is responsible for developing the features. They are characterized for being self-managing, self-organizing and cross-functional. The Team members are not only responsible for the success of each iteration but also for the project as a whole.
The Team can be formed by two until seven members. All of they are responsible to deliver the product. The Team comes up with the estimates, decide the user stories they accept to implement for the current sprint according to the priorities defined by the PO and report daily to each other. The Team is responsible for the "how" while the PO is responsible the "what". Teams characteristics are the following (Schwaber & Sutherland, 2017):

- Self-organizing;
- Cross Functional;
- There are no titles for Team members, regardless if a work is done by a person;
- There are no sub-teams in the Development Team;
- Individual members may be specialized in certain skills, but in the end, the accountability belongs to the Development Team as a whole.

The optimal size should be small enough to be adaptable but also large enough to complete the work within the Sprint. Small teams can encounter constraints, causing the Development Team

to be unable to deliver on time. Large teams create complexity that gets on the way of the process.

### Scrum Ceremonies

Scrum has four main ceremonies during an iteration that are used to establish regularity. All of them are time-boxed, being that all of they have a maximum duration. Iterations in Scrum are called Sprints - these are also fixed and should not be shortened or carried over.

Sprints duration may change and be adapted depending on the project, but the length should always stick between two to four weeks. The duration of the Sprint is short in order to maximize the collaboration between the Team and the Client. Having small iterations allows having more frequent feedback, minimizing risk and maximizing success. The Sprint duration should never be extended. In case of un-finished work on Sprint completion, the Team and the Product Owner decide if the work rolls-over to the next sprint or is no longer relevant and should be dropped.

At the Sprint end, the Team should have produced a shippable feature that brings value to the business.

During a Sprint, some rules must be followed, being the most important ones the Sprint Goal and the Team Composition. Scrum creates a safe environment where a Team can have predictable and fixed goals for a certain period of time. By changing the Sprint Goal, uncertainty is being introduced in the iteration, maximizing risk.

Within a Sprint, there are four ceremonies: Sprint Planning, Sprint Review, Sprint Retrospective, and Scrum Daily Meetings. Every Sprint has one meeting of each, with the exception of Daily Meetings that are held daily.

- *Sprint Planning*

Every Sprint starts with a ceremony called Sprint Planning. Just as the name states, is within this meeting that the Team, alongside the Product Owner and the Scrum Master, plan the next iteration.

The Scrum Master is the owner of this meeting, but the Team is the one deciding on the amount of work that they will commit to deliver. The Team compromises with the Product Owner on taking the most relevant work and completing it by the end of the iteration.

In this meeting, the presence of the Product Owner is of extreme importance since he is the one that is aware of the business priorities and has ownership over the Product Backlog. The Product Owner presents the features he would like to see developed and the Team divides the feature

into small pieces (tickets) and estimates the amount of work they will take. This estimation can be made using story points or ideal days.

When discussing a ticket, the Team should define the "Acceptance Criteria" to determine what marks the completion of the task. Having a set of criteria that the feature must reproduce/have is the best way to ensure work is completed with quality.

In order to keep the Team focused, this meeting should only take four hours (when the Sprint has two weeks) being that the duration can increase or decrease depending on the Sprint length.

By the end of Sprint Planning, the tickets selected and estimated are the Sprint Backlog, and this artifact is what the Team will focus on the next weeks.

Planning is crucial in the process of software development. Even though plans allow us to know critical information - "Who should be available to work on the project?", "Do we need more resources?", "Is this achievable?", "Is the project on track?" - planning is very difficult.

Planning is an attempt to find the solutions that optimize the product development. The team considers the features, the resources and the schedule they have. A good planning process will reduce risk and uncertainty, support better decision making and establish trust (Cohn, 2005).

- *Scrum Daily Meeting*

The Daily Scrum is a ceremony that occurs daily, always at the same place and hour, and should never be more than fifteen minutes. By keeping it short, the habit of having a quick meeting that does not affect the daily task is created, allowing for every member to understand the status of completion of the sprint.

During this ceremony, each team member answers to three simple questions.

1. What did I do yesterday to help with Sprint completion?
2. What will I do today to help with Sprint completion?
3. Is there anything blocking me from achieving Sprint completion?

The Daily Scrum is owned by the Scrum Master and it helps him understand how the Sprint is going and the velocity of the team. It is also a great way for the Team to be involved in everything that is going on and offering help in case of need.

- *Sprint Review*

The Sprint Review is held on the end of the Sprint. It is a place where the team can show the work done during the Sprint and get feedback from the Product Owner and from the shareholders.

The Sprint Review is an informal meeting, and the presentation of the deliverables is made with the purpose of creating feedback and enhancing collaboration. In this meeting various topics can be discussed, from future work that can be done, to bugs found that need to be fixed, and solutions to the implementation of new features.
The Scrum Master is the one making sure that this meeting is held and that the time box is respected.

- *Sprint Retrospective*

As the Sprint Review is for the product, Sprint Retrospective is for the process. In this meeting, the Team has the chance to inspect itself and suggest improvements for next Sprint.
The Scrum Master makes sure this meeting takes place and that it is productive. As it is stated on the Scrum Guide, the Sprint Retrospective has the following purposes (Schwaber & Sutherland, 2017):

- Annalise how the last Sprint went regarding people, relationships, process, and tools;
- Identify and order the major items that went well and what can be improved;
- Create a plan for implementing improvements.

### Scrum Artifacts

Scrum's artifacts are designed to maximize transparency on the communication of information so that everyone has the same understanding. An artifact is any metric that is able to show the progression of the product development.

- *Product Backlog*

The product backlog is a prioritized list of all the features needed in the product. This list has to be always available and ordered for the Team and the Stakeholders to use. This artifact is owned by the Product Owner that with the help of the Scrum Master keep it up to date. The product backlog has all the features, functions, requirements and bugs needed for future releases.
The product backlog is never complete, and it is in constant transformation. It evolves with the product and the environment it is surrounded by. The product backlog can be changed and re-prioritized at any moment because of its dynamic structure.
The Product Owner is responsible to collect all the requirements and features the client wants to have in his product. He takes the requirements and creates a list of features that need to be

developed, but since the PO is not a developer, he lacks the technical knowledge to turn the feature in actual work. Due to this, a backlog refinement is needed.

The process of backlog refinement is one to add details, estimate and break down features into user stories. A user story is a functionality that will be useful to the user and answers to three questions: who, what and why.

The user story goes through refinement in order to be polished and has all the necessary information, so any member of the team can just work on it. The Team is responsible to refine the user story and to estimate it.

Higher features in the product backlog mean that they are more important, are clearer and have more detail than the features on the bottom.

- *Sprint Backlog*

In Sprint Planning, the Team and the Product Owner go through a prioritized product backlog and decide on the piece of work that they can work on. At the end of sprint planning, the Team has a list of User Stories that they compromised to complete, called a sprint backlog.

The sprint backlog is owned by the Team, it is highly visible and should be a real-time picture of the work that the Team plans to accomplish during the Sprint.

- *Velocity*

The velocity is a measure of a Team's rate of progress. It is the sum of all story points that were completed in a Sprint. If a Team completes four user stories estimated with one, two, three and five-story points, their velocity is eleven.

If a product has all features listed in a Product Backlog and the Team estimates those features, the sum of the estimation is the size of the project. By using the Team's velocity, and by dividing the size by velocity we can reach a number of iterations necessary to deliver the product.

The velocity of a Team is based on historical data. By gathering the number of story points completed in each Sprint, we can reach the Team's velocity in the first few Sprints.

- *Burn-down Chart*

The burn-down chart is used to track the progress of the Sprint by plotting the number of days that are still remaining of the Iteration against the number of story points or ideal days still remaining (Agarwal & Majumdar, 2012).

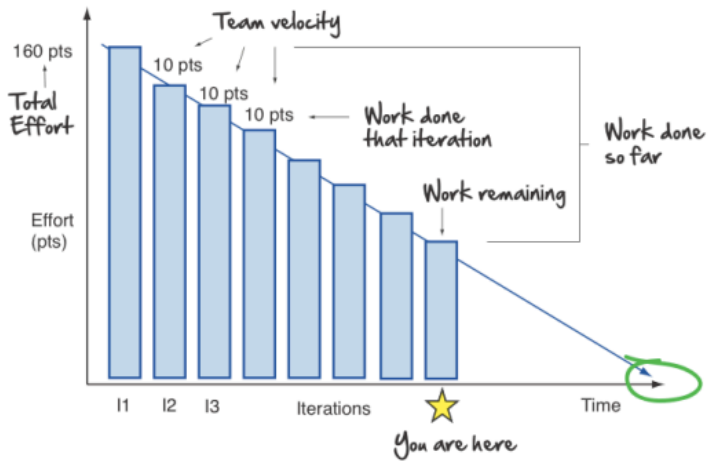


**Figure 4 - Burn-Down Chart**
(Rasmusson, 2019)

Burn-down chart is a two-dimensional way to show work remaining over time Figure 4. The y-axis represents the days left on a sprint. During the Sprint, every time work gets moved into Done, the graph goes down, until, ideally it reaches zero (Sutton, 2018).

Through one chart we can see the total effort, the team's velocity, the work done each iteration and the work that is still remaining.

## Scrum Flow

A Scrum project starts with a vision of the system to be developed. This vision can be from an external source like a client, or from an internal source.

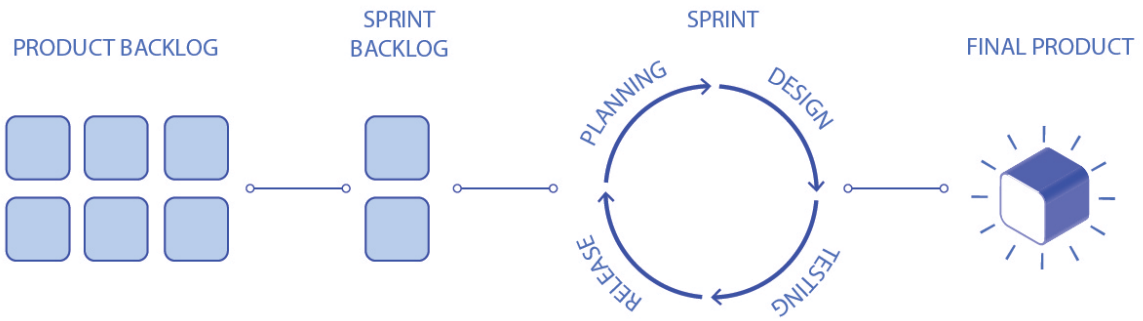


**Figure 5 - Scrum Flow**

In Figure 5 we can see the different phases of the process. Firstly, the Product Owner proceeds to understand the vision and transforms it into a list of requirements this product must have. This list of features is the Product Backlog and the Product Owner is responsible for it. This backlog, when turned into functionality will reflect the vision of the product.

The product backlog is prioritized so that the items that are more urgent to create value are a top priority, whilst others are a lower priority. Changes in the Product Backlog reflect changing business requirements and also how quickly the team can deliver the specific feature.

All work is done in Sprints. Before the beginning of each Sprint, the Product Owner holds a meeting with the Team and the Scrum Master called Sprint Planning. In this meeting, the list of features desired is presented to the team and the concept of the project is explained. It should be noted how important it is for the Team to have a broad understanding of the project, in order to be able to create a product with quality and minimize technical debt for future works.

After being presented with the prioritized Product Backlog, the Team discusses and makes a commitment with the Product Owner of delivering the next most important thing at the end of the Sprint.

In order to commit to a certain amount of work, the Team also discusses their capacity based on data from last Sprints (velocity, points delivered, person days) and based on the uncertainty and availability the Team Members will have for the next weeks. The Team is responsible for managing its own work. After Sprint Planning, the Sprint immediately starts, and the Team must do whatever is necessary to achieve Sprint Goal.

During the Sprint, the Team and the Scrum Master have Scrum Daily Meetings that last for fifteen minutes where each Team member answers three questions - "What did you do yesterday to help with Sprint completion?"; "What will you do today to help with Sprint completion"; "Is there anything blocking you from achieving Sprint Completion?" - to summarize the progress of the work.

At the end of the Sprint, the Team meets with the Product Owner, the Scrum Master, and the stakeholders to show what was developed during the Sprint. This is a way to let stakeholders understand the progress of the development process and to give feedback on the product being developed.

After Sprint Review and prior to Sprint Planning, the Scrum Master holds a Sprint Retrospective with the Team - the PO might also join this meeting when allowed by the Team - where the Team revises the Scrum framework and practices. In here, the Team talks about what went well, what went wrong and what can be changed in the future.

### Scrum estimating

Estimating in Agile differentiates size and duration, and both are estimated using different metrics - story points or ideal time.

It is important to understand that an estimate is a guess. The more time spent talking about a certain user story or on refining tickets to meet the definition of ready, the more accurate the estimation will be.

- *Story points*

Story points are relative and represent effort, not time. They are a measurement unit for expressing the overall size of a user story, feature or piece of work that needs to be done. Based on a study of 443 unique results, story points were perceived to be the most used form of estimation (Usman, Mendes, Weidt, & Britto, 2014).

In story points estimations, a raw value is assigned to a certain piece of work. In the end, the raw value by itself is not really important but rather the relative values. A piece of work that is assigned with the value two is twice as much effort as a piece of work with one story point.

The estimation takes into account the amount of effort involved in developing, the complexity of the implementation, the risk and also the dependencies a piece of work might have (Cohn, 2005).

In story points estimation, Teams use the Fibonacci sequence, in order to achieve more accurate estimations. The Fibonacci sequence is a series of numbers where a number is found by the sum of the numbers before it, starting by 0 and carrying on with 1,2,3,5,8,13 …

In the Fibonacci sequence, the gaps in the sequence become larger and larger as the numbers increase. This is helpful for estimating because as uncertainty on a user story increases, so does the inaccuracy of the estimation.

A user story should be broken into small tickets that can be refined and detailed enough to have small story points, in this way accuracy estimating increases and so does the predictability on the Team's velocity.

Agile teams are cross-functional meaning that they include all members required to build a product. When estimating in story points, teams learn how to better work cross-functionally, because they are to reach a single number to represent the whole team.

- *Ideal days*

On a software project the concept of ideal time - the planned time something should take - is different from elapsed time - the actual time something took - because of the overhead experienced every day.

In a working day is almost impossible to only do a certain task without being interrupted. Adding to the planned work one might answer to emails, making calls, interviewing candidates and be in meetings.

When estimating using ideal days, the Team assumes that the story being estimated is the only thing that a member will work on, that every single information needed is already provided and that there will be no distractions.
Estimating on ideal days is still a size estimation, although being less accurate than story points (Cohn, 2005).

- *Planning Poker*

In planning poker, the Team members are given a deck of cards that has written on it a validate estimate.
The moderator that can be the Product Owner, an Analyst or any team member, reads the user story or the details on a ticket that needs to be done. The Team asks questions, gathers all information necessary. After all, questions are answered, each Team member secretly chooses a card that reflects their estimation on it. The cards are only shown when every member has decided and picked one. After so, all cards are simultaneously turned over and shown so that everyone is aware of the estimation.
It is normal for estimations to be different. If that is the case, the Team member with the highest and the lowest estimation explain what their train of thought was and what made them choose that value. This discussion is important because it helps the team oversee some difficulties that they may have not thought about or simplify work that seemed to demand high effort.
The group proceeds to discuss their estimations for a couple of minutes, and after that proceeds to re-estimate based on the new information obtained. The process is identical to the one in the beginning. In most cases, estimates will converge in the second round but if so does not happen, the process is repeated until the Team reaches an agreement. The unanimous agreement is not always possible, so in the case of having a higher number of identical estimations over a lower number of different ones, the moderator asks if they are happy to go to the most common estimation (Cohn, 2005).
Planning poker brings together multiple expert opinions that form a cross-functional team. Team members need to be able to justify their estimation, improving accuracy and helping complete some information that might not have been thought about.

# 3. CASE STUDIED: LIVING MAP

For the case study, there will be a full description of the company where this research was held and the use cases that are worked on the company. Following that, there is a SWOT, a PEST and a competitor analysis.
In order to better understand the company's structure, it is presented an organisational diagram and a brief explanation of the three different areas within the company - User Centred Design, Geographic Information System and Software Development. The interaction between these three areas and the map creation is detailed after, as well as the structure used in JIRA and their workflow within the company's system.

## 3.1 Living Map

Living Map is a SaaS (Software as a Service) solution that provides an integrated product for map management and delivery to consumers. This is achieved through browser-based digital maps as well as with native mobile app components.
Living Map is currently developing its own unique indoor positioning technology, to provide accurate location and navigation services in any environment. The positioning system that is being developed will eventually provide to third-party integrators as a Software Development Kit (SDK), uses sensor fusion technology and combines a variety of datasets taken from the device as well as external inputs such as Wi-Fi scanning, magnetic data, and a pedestrian dead reckoning algorithm.
Living Map bridges the information gap between people and places, creating better user experiences and increased operational efficiency. Maps are the perfect interface for understanding any environment, and digital map technology has emerged as the essential enabler of multiple consumer and business applications, from navigating cities to visualizing live asset data generated by the Internet of Things sensors.
Living Map works with many different use cases:
**Cities**: a city can be a challenging environment to navigate for both residents and visitors, who need a tool to support them during their journey and provide them an overview of what the destination has to offer. One of the latest trend spreading globally is about Smart Cities, meaning those places that use innovative technology to enhance the quality of life for the people.

**Heritage:** museums and art galleries do not usually have recurrent customers, so every day they are presented with the challenge of helping new people navigate their environment. Wayfinding around multiple rooms or buildings, and artworks location can be difficult for visitors, to provide people with the information they need to locate a specific asset and navigate there would save the venue a lot of time and efforts.

**Venues**: venues such as shopping centres, arenas, and stadiums are trying to meet user's needs by transforming their digital experience, and assuring them recommended accessibility standards, improving the customer service and make events more appealing. A seamless experience is becoming essential for customers, when either they're trying to find their seat, a specific retailer, or simply understanding food and drinks offer at the venue.

**Transport**: digital transformation is an essential strategy for any company operating in today's travel industry. Train stations, bus station, airport, and multimodal stations have recently been showing higher attention to customer service, passenger experience, and digital environment. Travelers are adapting to change to the new digital environment surrounding them, and this reflects also in their habits.

**Healthcare**: healthcare organizations make use of Internet of Things (IoT) technologies to improve their workflow and patient experience by trying to meet new patients' demands, to capture and analyse data. A significant fraction of missed hospital appointments is due to navigation problems and hospital staff think that wayfinding apps could increase their efficiency, as they will not be required to act as tour guides to hospital visitors. The practice of medicine is being reinvented, and hospital navigation and asset management are part of it.

**DOOH**: digital out-of-home advertising is that kind of advertising that reaches consumers while they're outside of their homes. Dynamic media located in specific areas of a public space are not only a valid marketing opportunity for companies, but also an opportunity for cities or venues to share information with people. To make people look at those screens, kiosks and interactive media, they should contain interesting and attractive content that may be useful to them.

## 3.2 SWOT analysis

Table 1 - Living Map SWOT analysis

| STRENGHTS | OPPORTUNITIES |
|---|---|
| - Team of resourceful people<br>- Design expertise<br>- Open platform | - Growing digital map market<br>- Smart Cities projects<br>- New user experience<br>- Data regulation |
| **WEAKNESSES** | **THREATS** |
| - Immature product<br>- Indoor positioning system<br>- No specific vertical/market | - Infrastructures are changing<br>- Privacy/GDPR<br>- GPS doesn't work indoor<br>- Competitors |

Living Map is a team of young and resourceful people, ready to work to meet client's needs and to provide them with the best solution possible. A unique blend of geospatial experts, cartographers, engineers, and designers combine to transform the way people experience the cities, buildings, and businesses around them.

Living Map is led by a team with rich experience in urban design and smart city technology. Understanding how people interact with a map and how they move in a complex place is an essential part of building a user-friendly design to ease people's experience.

Living Map is an open platform that can connect with third-party technology provider and IoT sensors, to create custom functionalities and retrieve real-time data for a more general and clearer overview of a site.

Despite the effort put in the company in 2017, the product is still immature and needs to be defined more clearly. Some of the features are now standard and can be delivered to every client, but there are some features that should have been developed by now, and still are not. Until those are ready, Living Map will not have a full and complete product to sell to clients, but it will be a project in development with them, over time.

Living Map Indoor Positioning System is still under development, and unfortunately, until it is ready, there will be something missing in the product offered to the clients. The ability for a user to see his accurate location reflected on the map is becoming essential to every service provided, so some effort needs to be put in the development of this technology, to be finally ready to meet customer's expectations in that field too.

Living Map still does not have a clear and specific target market. Living Map defines its target as every 'complex place' but the term is quite broad, so it needs to be refined to a smaller number of verticals to focus on.

Living Map can be considered to operate in two different markets: Digital Mapping and Indoor Location. The forecast for both markets is of positive growth.

Smart Cities are growing as a sustainable solution for urban life: renewable energies, smart grids, intelligent traffic control, electronic government, urban mobility, etc. All the major cities around the world are trying to become "smart". They have a development plan concerning these new problems and they want to use the data they collect in the best possible way.

This new way of mapping can change the way how retailers and shops market to clients. Making store locations easier to navigate presents obvious benefits for both shoppers and retailers, improving customers' experiences, and streamlining sales.

Recently more and more businesses are afraid of giving away their data, they still want to own it. With the Internet of Things and the Cloud, it is becoming more challenging for users to own their data, but Living Map provides a secure data storage system, so every single client will still own their own data, that will not be used by Living Map for any other purpose if not developing their platform.

Infrastructure is always changing, shops are opening and closing all the time, so the maps need to be constantly updated and this requires time and money. Living Map teams are trying to optimize this phase as much as possible, but unfortunately, at the moment it still requires quite a lot of work and time. Once the system will be switched to vectors and the optimizer will be ready, everything will be much easier.

Location data reveals a lot about a person. It can be telling about the home and workplace of a person, but also even more sensitive data, like religion or political activities. Some sets of data can be amended in such a way that no individuals can be identified from those data (whether directly or indirectly) by any means or by any person. Ensuring that there is no way in which individuals can be identified is a technically complex task. Living Map (LM) will need to discuss how user's location data is processed and LM role when the SDK is implemented into 3rd party apps.

A Geographic Positioning System (GPS) requires a satellite line of sight and doesn't work well indoors, finding a valuable alternative requires time and money. There are multiple alternatives to GPS to retrieve the location indoor, but this industry is still very young, and a lot of questions are still unanswered.

Living Map competitors can be identified in Indoor Mapping companies, Location and Navigation companies, Open Source Map companies, Geocoding companies, and even Data Analysis companies. For a clearer overview, please see the following section.

## 3.3 PEST Analysis

Table 2 - Living Map PEST analysis

| POLITICAL | ECONOMIC |
|---|---|
| - Intellectual Property Rights<br>- Non-disclosure agreement<br>- General Data Protection Regulation (GDPR)<br>- Brexit | - Exchange rates<br>- Indoor location/navigation market |
| **SOCIOCULTURAL** | **TECHNOLOGICAL** |
| - Lifestyle trends<br>- Urbanisation<br>- Out of Home (OOH) advertising | - Competing technologies<br>- Impact on cost structure |

Protecting intellectual property makes it easier to take legal action against anyone who steals or copies it. There are different types of protection, depending on the intellectual property. Living Map can claim protection on the design of each map released. Living Map Proprietary Positioning System can be protected by external risks, only if covered by a patent.

Before starting any business with other parties and sharing sensitive information, it's safe for both parties to sign a non-disclosure agreement, to make the relationship confidential and to protect any type of confidential and proprietary information or trade secrets.

The new General Data Protection Regulation for European Union members is about to be released, and it will unify data protection law across all member states, describing requirements for companies and organizations on collecting, storing, processing and managing the personally identifiable data of their employees and customers. To respect this new regulation, Living Map will have to ensure that privacy notices are present wherever personal data is collected, and it will be needed to discuss how user's location data is processed and LM role when the SDK is implemented into 3rd party apps.

Brexit will most certainly start its effects in 2019, and it will affect trade across the European Union. Between the impacts that can already be seen at the moment, we can find the pound currency value that reached the lowest level in 31 years against the dollar. Another effect is the inflation that reached its fastest pace in four years, while economic growth has slowed. For what concerns foreign workers, no agreement has been made yet, so no one knows how it's going to be. Uncertainty is still high at this point, but one thing we can be sure about is that in the short term, UK regulatory law will remain connected to the EU regulations.

Dealing with companies in other countries, Living Map has to face the problem of currency exchange rates. Most of LM clients are in the US, UK, rest of Europe and United Arab Emirates

(UAE), so the currencies to keep an eye on are mostly Euro and US Dollar. Today's rate (30th, January 2019) is 1.32 US$ and 1.16 Euro. One way to be 'safe' in all negotiations could be costing everything in Pounds anyway, even if the client has a different currency. In this way we'll be covered in case of a value decrease.

Smart homes, tech gadgets, and virtual assistants are now part of everyday life and everyone is being affected by them. Related to maps, the trend is definitely knowing what's going on around you in real time and being able to visualize your own location on the map, wherever you are. The main trends developing in the market will be Location Based Services and Personal Digital Assistants, giving the user the ability to access information from online sources. In 2050, there will be an additional 2.5 billion new people in the world - most of which will be living in urban areas. Driven by increasing economic globalization, urbanization levels are anticipated to reach 66%. Beyond the positive environmental impacts that technology can drive - such as LED streetlamps, energy-efficient buildings, and the Internet of Things (IoT) - based products - it can also have significant social benefits in an increasingly urbanized world (Technavio, 2017). Outdoor advertising influences consumer behaviour more than the advertising at home, lately. The research found that people who have seen an out-of-home advert are 17% more likely to interact with the brand. The opportunity here lies in collaborations with DOOH screens and kiosks providers.

An Indoor Positioning System (IPS) is a system to locate objects or people inside a building using radio waves, magnetic fields, acoustic signals, or other sensory information collected by mobile devices. There is no standard for an IPS system, but the number of companies entering this industry and challenging themselves with different technologies is growing. Pedestrian Dead Reckoning (PDR), Wi-Fi, Beacons, Landmark Beacons, and GPS, are all technologies available to everyone, so Living Map needs to make sure to develop a system that blends all those inputs in a different way, to be able to differentiate themselves.

## 3.4 Competitors Analysis

As specified in the SWOT analysis, Living Map still does not have a specific target market, and for this reason, it is also difficult to recognize one specific segment of competitors. They can be either Indoor Mapping companies, Location and Navigation companies, Open Source Map companies, Geocoding companies, and even Data Analysis companies.
There are two main industry categories where it's possible to find Living Map's competitors - Digital Mapping and Positioning System.
Most of them are located in the United States, but there are also some European companies in this business. Here below there is an overview of a few of the competitors in both areas.

### 3.4.1 Digital Mapping

- Esri: Esri is a Californian data analysis company that works with maps on a large scale. “Esri pursues mapping and spatial analysis for understanding our world with visionary products and services that define the science of GIS”(ESRI, 2019). They use geoscience to solve problems and help organisations create positive change in industry and society.
- LocusLabs: LocusLabs is a Californian company that operates in the airlines industry. They developed a mobile based indoor location platform and provide venue maps all around the world. They say their mission is “to help people find whatever it is they are looking for”(LocusLabs, 2019). They are currently quite spread in the US and thanks to the collaboration with airlines, they’re starting to reach European airports too.
- Mapbox: Mapbox is a Washington-based company that provides city maps. Since they are a provider of custom online maps that the user can customise by himself, they describe themselves as “the location platform for developers and designers” (Mapbox, 2019).
- Micello: Micello is a Californian company that operates in the Indoor Maps market that provides maps and data to enable indoor location-based services.

### 3.4.2 Positioning System

- Accuware: Accuware is an American company, based in Florida, built to deliver turn-by-turn indoor navigation throughout a venue. Their system blends ambient

Wi-Fi and iBeacons signals with data from devices' internal sensors to deliver devices' locations with sub-meter accuracy (Accuware, 2019).

- Cisco: Cisco, the American multinational technology conglomerate, has developed a product called Cisco Hyperlocation to deliver location accuracy for indoor Wi-Fi. Location data is combined with Bluetooth Low-Energy (BLE) beacons and sensors on the mobile device, for near real-time navigation and location services.
- IndoorAtlas: IndoorAtlas is a Finnish company that provides an indoor positioning system to locate people and objects inside a building using radio signals, geomagnetic fields, inertial sensor data, barometric pressure, camera data or other sensory information collected by a smartphone device or tablet.
- Pointr: Pointr is a British company specialised in "indoor positioning of people, assets and deep data analytics" (Pointr, 2019). Their real time indoor positioning system doesn't rely on phone's compass, which in many indoor environments fails to deliver useful orientation, but instead on beacons and inertial sensors that they position on site.

## 3.5 Organisational Structure

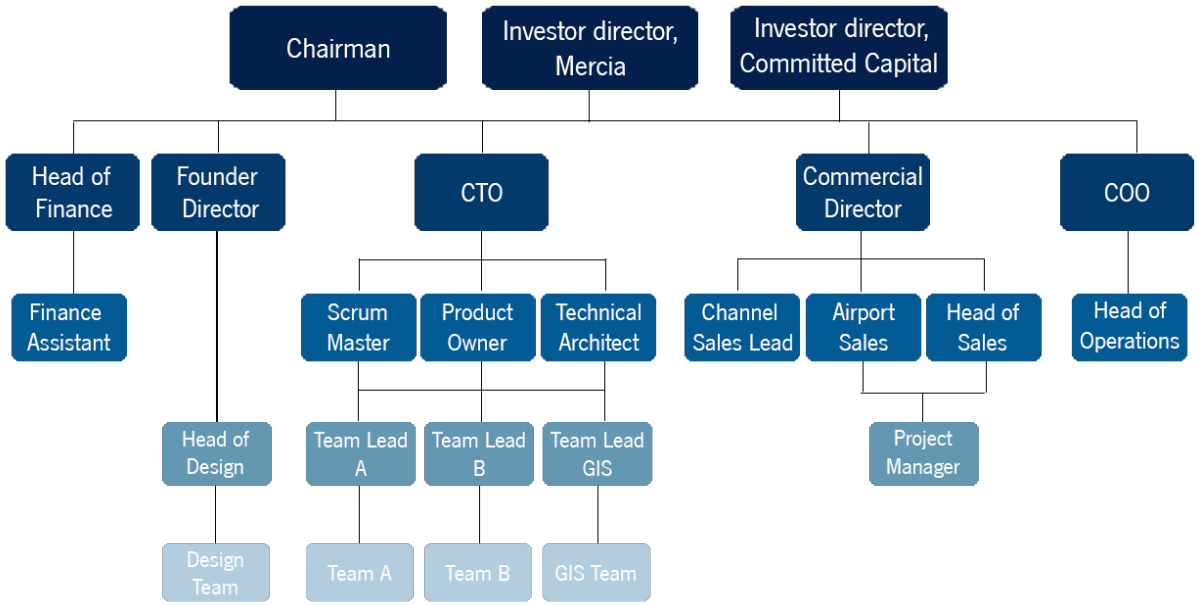


Figure 6 - Organisational Structure

Living Map currently has 34 employees in two big cities in the United Kingdom – Bath and London. In Figure 6 we can see that there are two main investors that work alongside the

Chairman. The board meetings involve these three roles plus all of the directors - Head of Finance, Founder Director, Chief Technology Officer, Commercial Director and the Chief of Operations.
The Founder guides the Design Team, composed by the Head of Design, a Cartographer, a UX designer, an Artworker and a Graphic Designer.
The Chief Technology Officer is responsible for three teams - two development teams and one GIS team. The Scrum Master and the Product Owner are a part of these three teams, alongside the Technical Architect. Each development team has 7 members, and a tester that is responsible for both of the team's test. The GIS team has 3 members.
The Commercial Director has a sales team with three members that focus specific areas. Both Head of Operations and Head of Finance have one member on their team.

## 3.6 Areas

### 3.6.1 Geographic Information System

Geographic information systems (GIS) is software that provides a diverse range of functions to create, acquire, integrate, transform, visualise, analyse and archive information about the surface of the earth. GIS associates locations in space with variables like temperature, population density, land use or elevation (Goodchild, 2006).
Geographic information is broken into three components (Chrisman, Wiley, Chichester, Toronto, & Weinheim, 1999):

- Space: our work is three-dimensional, objects have length, width and height. GIS analyse and interpret the three-dimensional shape of the earth and transform it into the two-dimensional maps.
- Time: most maps are a snapshot for a specific moment it time.
- Attribute: can go from what is physically observed to aesthetic judgements. The attribute value is a characteristic related with a geographic feature.

Whilst time and space have standardised systems - Temporal Reference System and Spatial Reference System - attributes vary the most.
By relating what it seems to be unrelated data, GIS helps individuals and companies to understand spatial patterns. GIS also uses location as an input for information. This input can be expressed in latitude and longitude or as a ZIP code.

GIS can use various types of data such as population, income, education level, landscape, different vegetation, soil, roads, electric power lines, etc. The data is used so users can compare the locations of determined things and discover how they relate to each other.

- GIS uses hardware and software systems. The data can come in cartographic or photographic data, digital or even on data spreadsheets.
- Cartographic data already is a map format. It has information like the river location, roads and valleys.
- Photographic interpretation is a very big component of GIS. The process involves analysing photographs and assessing the features that are represented there.
- Digital data is also used in GIS. A satellite data can be used to show land.

GIS takes data that might come in different forms and overlays it on top of another in a map. The process of putting information into GIS is data capture.
The most common types of GIS file formats are raster - grids of cells that are useful for data that may vary like elevation or satellite imagery - and vector - polygons that used points, also called nodes, and lines.
Once all the data is gathered into a GIS system, they are combined to produce individual maps. These maps can show if houses are in areas prone to flooding, earthquakes or volcanic activity. GIS can also help researchers study the changes over time. By using satellite data, they can analyse topics such as ice cover in polar regions and monitor how it changes over time.

### 3.6.2 User Centred Design

A lot of software systems have the end user - people. A system should be easy to learn, to use, as well as be functional, so the design of a software is a very important step in the process.
There are three main principles of design (Gould & Lewis, 1985):

- Early focus on users and tasks: where the end user is clearly identified. The team studies the requirements and tries to understand what the end goal of the software is. Who is it for? What should it do? This process should be user driven;
- Empirical measurement: after designing the wireframes, the users should actually test through a simulation, what the system is meant to do. The user's performance and reactions are observed, recorded and analysed;
- Iterative design: problems found in the user testing have to be fixed.

The design cycle is iterative, so the stages of design, testing and measurement are repeated and if needed re-designed, as many times as required.

User Centred Design (UCD) is a term used to describe the design process where end-users have a say on how the end design looks. The design process is built upon participatory action research. This research involves the participant identifying usability problems, which are then resolved by the designers. Don Norman mentions four principles on how a design should be (Norman, 2013):

- Make it easy to determine what actions are possible at any moment;
- Make things visible, including the conceptual model of the system, the alternative actions, and the results of actions;
- Make it easy to evaluate the current state of the system;
- Follow natural mappings between intentions and the required actions; between actions and the resulting effect; between the information that is visible and the interpretation of the system state.

The role of a designer is to facilitate the task and make sure that the end product functions as it was intended with minimal effort for the user. In order to make difficult tasks simpler, Norman also says that a designer should (Norman, 2013):

- Use both knowledge in the world and knowledge in the head;
- Simplify the structure of tasks. The user is capable to remember five things at a time. The task should be consistent and have mental aids to easily retrieve information from Long-term memory;
- Make things visible: bridge the gulfs of Execution and Evaluation;
- Get the mappings right;
- Exploit the power of constraints, both natural and artificial;
- Design for error;
- When all else fails, standardise.

### 3.6.3 Software Development Process

The software process is all the activities that lead into the production of a software product. These activities can include developing something from scratch, modifying existent systems or software integration.

There are four fundamental activities to software engineering (Sommerville et al., 2011):

- Software specification where the details about the product are defined;
- Software design and implementation where the product is produced;
- Software validation where the software is tested and validated;
- Software evolution where the product is upgraded to meet what the customer needs.

Besides these four activities there are also sub-activities like requirements validation, architectural design and unit testing, as well as supporting activities like documentation and software configuration management.

### Software Specification

The software specification is the process of collecting, analysing, understanding and refining what features and services are required from the system and identifying what are the constraints for the development process.

This is a crucial step in the process because errors at this stage will inevitably cause problems in future stages. We can take as an example the following: a business developer meets a client and shows him a map of a city for data visualization. This map has a certain styling and design. The client tells the business developer that he wants the same map, so the business developer talks with the Team, that starts to create a new map with a new design and stylesheet. When showing the final product, the client is extremely disappointed because he wanted the first design and stylesheet and does not like the new one. The team has to refactor all the work because the client's expectations and requirements were not correctly gathered. This leads to over-work and money lost.

Breaking it down to the basics, a software requirement is a property that something must possess in order to solve a problem present in the real world (IEE Computer Society, 2014).

The requirement engineering serves to produce an agreed document with all the requirements that the system must have.

There are two types of requirements for a software to be developed – a process requirement (that lays a constraint on the development of software) and a product requirement (a need on the product to be developed) (IEE Computer Society, 2014).

These requirements have two level of details - a general detail for the end users and customers and high details for the system developers. Ian Sommerville identifies four main activities in the software specification stage (Sommerville et al., 2011):

- Feasibility study – this study should be low in cost and time. This study has as a main goal understanding if the product will satisfy user needs, if it will be cost-

effective from a business point of view and if it is possible to develop it based on the existing constraints. The result is the decision to go ahead or not with developing the product.

- Requirements elicitation and analysis – existing systems are analysed in order to derive the system requirements out of them. It may take discussions with potential users, task analysis, use case tests or the development of a new system with models and prototypes.
- Requirements specification – translating all the information gathered into a set of requirements. This includes the user requirements, that are abstract statements and the system requirements that are detailed with the description of the functionalities required.
- Requirements validation – checking if the requirements are achievable, consistent and complete.

These activities are not carried in a strict sequence and are rather a continuous process – new requirements might appear during the overall process.

### Software design and implementation

During the software design and implementation, the system specification is converted into an executable system. A design can be thought as a form to solve the problem which is a very important part on the software development process (IEE Computer Society, 2014).

By the end of the software design, there should be a description of the structure of how the software should be implemented, as well as the interface of the components and the algorithms used. The design takes as input the platform information, the requirements specification, and the data description and uses them to create a system architecture, database specification, interface specification and component specification. Design process activities are interleaved based on the feedback from one stage to another and inevitably rework in all the process.

Depending on the system being produced, design activities vary, but four of them are (Sommerville et al., 2011):

- Architectural design, where the structure and the components are identified as well as their relationship;
- Interface design, where the interfaces between systems are laid. These interfaces need to be very detailed and specific;
- Component design is the activity that explains how each system will operate. It may be automatically used to generate an implementation;

- Database design where the system data structure is designed and shows how it will be represented in a database.

After the design process, the development takes the design outputs. Software development tools may be used to originate a skeleton program from a design.
There are four fundamental constructions that should be followed (IEE Computer Society, 2014):

- Minimising complexity – achieved through creating code that is simple and easy to read.
- Anticipating change – since software changes frequently, anticipating changes helps the engineers build extensible software;
- Constructing for verification – building software where faults can be easily found by testers or other software engineers;
- Reuse – where the same asset can be used to solve different problems in different situations;
- Standards in construction – in order to achieve a project's objectives for efficiency, quality and cost, development standards should be applied during construction.

Programming does not have an exact process because it is a personal activity and depends on what it works better for the developer. Per the norm, developers have some kind of test on the code they developed in order to identify defects that need to be fixed. Testing helps to identify bugs, whilst debugging is the process of localizing the bug and fixing it (Sommerville et al., 2011).

### Software validation

Software validation can also be called as the verification and validation process and it is where it's confirmed that the system meets the specifications and goes towards the client expectations. One of the principal validation techniques is hard-coded data into the system. Due to the predominance of testing, most validation costs occur during and after implementation (Sommerville et al., 2011). The stages in this process are:

- Development testing where the developers test the system. Each component should be tested independently from other components.
- System testing where all the components are integrated to form the complete system. This will unravel bugs from the interaction of systems with one another. It is also meant so the system can display the non-functional and functional requirements.

Accepting testing where the system is tested with data supplied by the system costumer rather than being used hard coded data.

### Software evolution

It is rather easy to make changes to software when in comparison to hardware. In the book Software Engineering, Ian Sommerville says that most software systems are a re-use of already existing ones (Sommerville et al., 2011).

This means that the need for maintenance is quite high and sensible. Software is always changing and can always be upgraded over a lifetime in response to changing requirements, new requirements and customer needs.

The objective of maintenance is to modify software that already exists while still maintaining its integrity. There should also be regular maintenance tasks prior to delivery of software (IEE Computer Society, 2014).

## 3.7 Creating a map

Creating and customizing a digital map is not as easy as it may seem. It requires expertise in multiple sectors, and constant monitoring and maintenance. The three areas of knowledge are, Geographic Information System (GIS), graphic design and UX design, and software development.

The first step to map creation is to analyse the data received. The GIS team receives cartographic or topographic data about the area to map. This data can come in diverse format styles.

After going through the data and deciding if it is usable, a process called georeferencing starts. Here, GIS has to connect the coordinate system of the map to the corresponding position on the surface of the earth (using latitude and longitude coordinates).

The next step is mastering the process of creating a point, line and polygon data to create a map database. Here is where the team draws the map with all the information that will be needed.

Once the base map is ready, the GIS team creates a database, a structured set of data, safely stored and connected to software that can query the data and visualize it on the map. This database differs from a normal database because it includes spatial locations and shapes connected to the map.

In a map, all the assets (e.g. rooms, coffee shops) are represented by a polygon directly connected to the database where the software retrieves all the relevant information the client wants to show and have visualized on the map.

When a project starts, the Product Owner has a list of specific requirements that is delivered to the Design Team. They will analyse and develop the best User Interface (UI) and User Experience (UX) for the client. In this phase, a list of wireframes and prototypes are produced to be presented to final approval and used as a guideline during the map development.
After the final design is agreed upon, the design team work on a stylesheet that gathers the structure of the map - fonts, colours, icons - that will be used for the digital version of the map. The next step is up to the development team that will follow the design and GIS guidelines and make the map go live.

## 3.8 JIRA software

Living Map uses JIRA, a project management tool, owned by Atlassian. This is a management tool that helps in Scrum organization. Here, the Product Owner and the Business Developers create tickets (user stories) that are connected to epics (features) and to versions (deliverables), all in one place. It also enables the virtual visualization of Scrum boards, as well as Scrum artifacts.
All members of the Team have access to the Scrum board, to the tickets and to the historical data collected.

### 3.8.1 Geographic Information System

GIS uses a Scrum board with six columns. As we can see in Figure 7 those columns are – “To Do”, “In Progress”, “Blocked”, “Awaiting in Design”, “Questions and Answers (QA)” and “Done”. When starting a Sprint, the Sprint Backlog is displayed in a column named “To Do”. The Sprint Backlog can have:

- GIS stories: pieces of work that must be done to achieve a certain Epic;
- GIS bugs: problems found and reported that should be fixed.

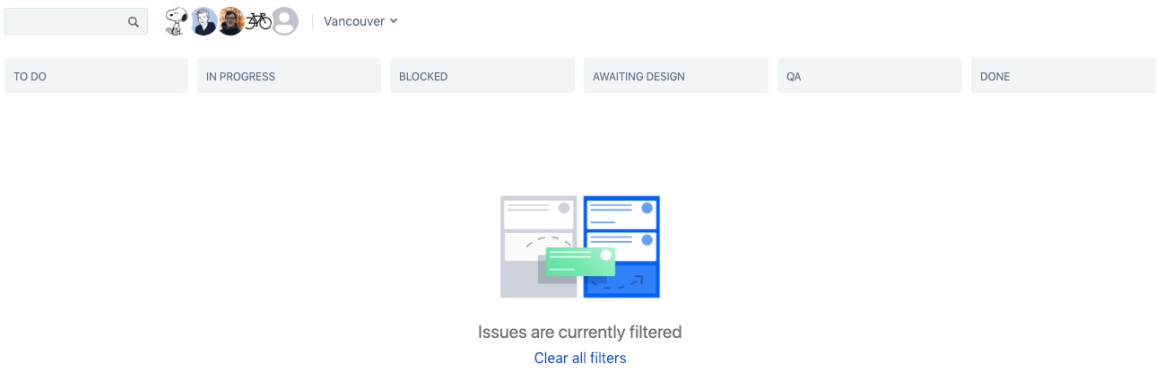


Figure 7 - GIS Sprint Board

The column "To Do" is prioritised by the Product Owner so tickets that are more important are on the top of the column. When the Sprint starts, and a Team member starts working on a GIS story, the correspondent ticket is moved by the member to the next column, called "In Progress". After all the work on the ticket is done, the Team member moves it to the column called "QA" where another member of the Team will revise the work done and approve it or reject it. If the ticket is not approved, it goes back to the column "In Progress" and if it passes "QA" it is moved to the column "Done".

Between "In Progress" and "QA" there are two other columns called "Blocked" and "Awaiting Design". These two columns are very similar, with the only difference that a blocked ticket is blocked by an external dependency whereas a ticket in "Awaiting Design" is blocked by the Design Team.

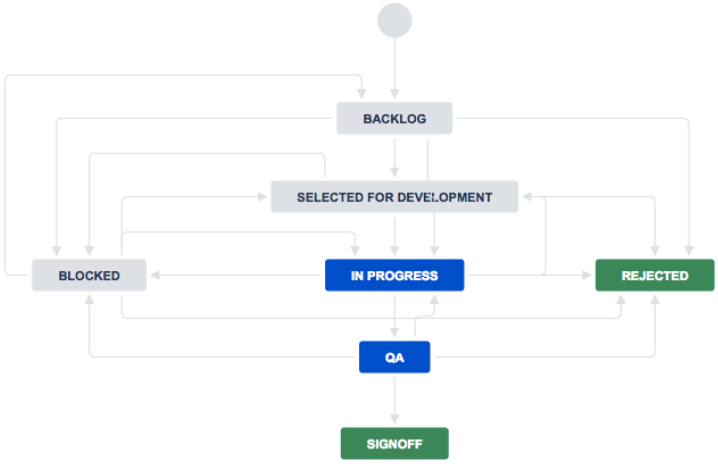


Figure 8 - GIS Ticket Workflow

The ticket workflow within the Sprint board is a little bit more complex. As we can see in Figure 8, when a ticket is created, it starts the workflow in the "Backlog". By following a normal flow, the ticket would be "Selected for Development" by moving in to the "To Do" column – when the sprint starts. When the GIS engineer starts working on that ticket and moves it to "In

progress", the ticket gets the same status in the workflow. After the work is done, it would get the "QA" status, and after that the "Signoff" status, when the ticket is moved into the "Done" column.
A ticket can be moved to and from the "Blocked" status at any given point in the workflow.
The workflow also allows for a ticket to be "Rejected" when:

- It is in the Product Backlog, never being moved into the Sprint Backlog;
- It is in the Sprint Backlog, with the status "Selected for Development", "In Progress", and in "QA". These statuses also allow for an "Un-reject" flow.

### 3.8.2 Software Development

The Development Team is divided by two teams – Team A and Team B. These teams share two types of boards in JIRA destined for its tickets. One is a Kanban board called "Refinement board" and the other is a Scrum board where two different sprints are shown in order to maximize transparency. As we can see in Figure 9Figure 9 - Software Development Sprint Board, the sprint board has 8 columns, "Dependent/Pending"; "To Do"; "In Progress"; "Blocked"; "Ready for Code Review"; "Code Review"; "Passed Code Review"; "Ready for Testing"; "Testing" and "Done".
When starting a Sprint, the Sprint Backlog is displayed in a column named "To Do". The Sprint Backlog can have:

- Stories: pieces of work that must be done to achieve a certain Epic;
- Bugs: problems found and reported that should be fixed.

The column "To Do" is prioritised by the Product Owner so tickets that are more important are on top of the column. When the Sprint starts, and a Team member starts working on a Story, the correspondent ticket is moved by the member to the next column, called "In Progress". The member works on the ticket, and once all the Acceptance Criteria is met, the member moves the ticket to "Ready for Code Review".
The Team members know that when they finish a ticket, they should first see if a ticket is in "Ready for Code Review" before starting a new ticket. When a member is doing a code review, the ticket should be moved to the column "Code Review". If a ticket passes the code review, it is moved to the "Passed Code Review" where the last piece of work (deployments) are made before moving to "Ready for Test". If the ticket does not pass code review, it goes back to "In Progress".

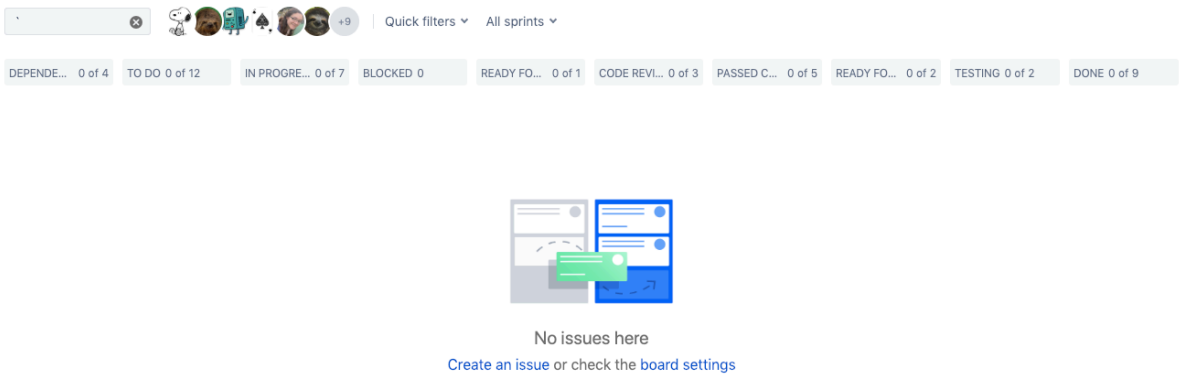


Figure 9 - Software Development Sprint Board

After the tester starts testing the work, the ticket has to be moved to the column "Testing". If the ticket passes the test, it moves to the "Done" column, whereas if it does not pass the testing, is moved "Into Progress" once again.

When a ticket is created it goes to a "Refinement Board" in order to be detailed until it meets the **Error! Reference source not found.**, so it can be estimated and considered as an item in the Product Backlog.

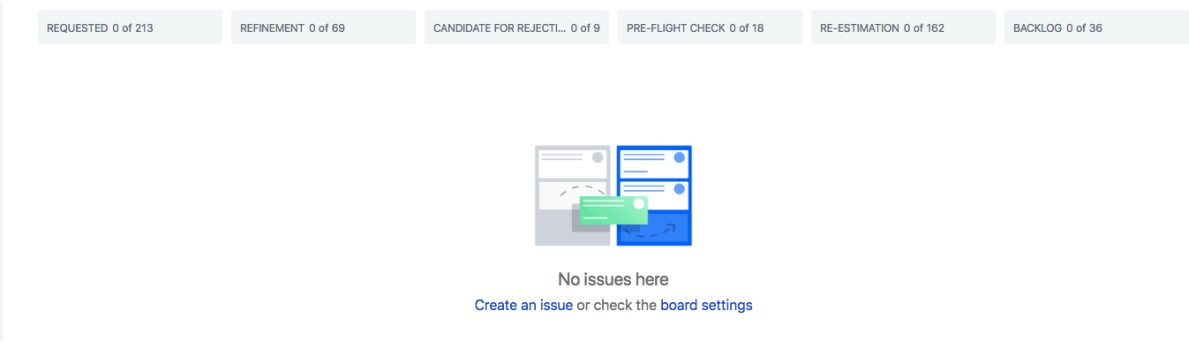


Figure 10 - Software Development Refinement Board

The Refinement board has 6 columns – "Requested"; "Refinement"; "Candidate for Rejection"; "Pre-Flight Check"; "Re-Estimation" and "Backlog", shown in Figure 10. As soon as the ticket is created, it goes into the "Requested" column. Before the Backlog Refinement, the Scrum Master and the Product Owner have a quick meeting where they move the relevant tickets into the "Refinement" column.

In the Backlog Refinement, the Teams go over the tickets that are in the "Refinement" column. Here, specific detail about the implementation of the ticket is added and some questions are raised in order to fully understand the piece of work that needs to be done.

After the Backlog Refinement, the tickets that were discussed are moved to the "Pre-Flight Check" column, where they sit, until the Ticketing meeting, where they will be estimated and be Ready to move into the "Backlog" (also known as Product Backlog).

If a ticket is created and not touched for a very long time and stops being relevant, it is moved into the “Candidate for Rejection” column, where it will be rejected after a period of time. If a ticket has been in the Backlog for a very long time, it is moved into the “Needs Re-estimation” column where it will be reviewed and discussed.

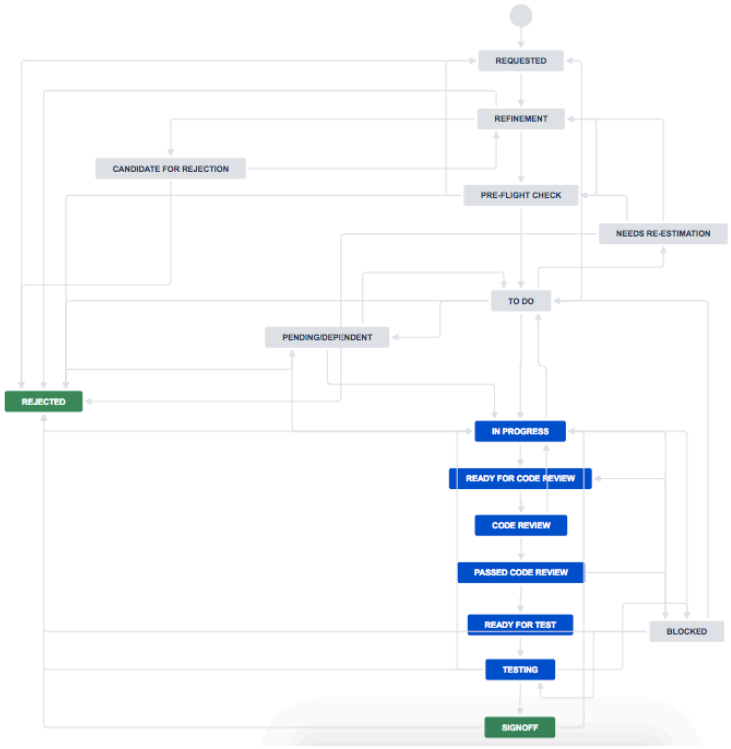


Figure 11 - Software Development Ticket Workflow

The ticket workflow within the Development Team is complex due to the fact of having two boards connected – the Kanban Board for “Refinement” and the Sprint Board for the Sprints. In Figure 11 we can see the diagram that represents all the statuses a ticket can pass through.

Following the normal workflow, a ticket would be created and put into the “Requested” column and have the same status as the column. It would follow the process, passing through “Refinement”, “Pre-Flight Check” and in the end being moved to the “Backlog” column in the board, having the status “To Do”.

When a ticket has the “To Do” status, it means the ticket passed to the intermediate place between the Refinement board and the Sprint Board. These tickets meet the “Definition of Ready” and are candidates for the Sprint. When in this place, the ticket can go back to the “Refinement” board if needed, by moving it into the “Needs Re-Estimation” column. This happens when a ticket passes to the backlog, but the Team feels like the ticket should be re-estimated due to changes in the description, environment or circumstances. When in “Needs Re-Estimation” a ticket can be re-estimated and moved back to the “Backlog” column with the “To Do” status, or it can be moved in to the “Refinement” column if in need of refinement, or

to the “Candidate for Rejection” if the Team feels like the ticket was over-seeded by another ticket or it is no longer relevant.
When the ticket is moved to the Sprint Backlog, it goes through the normal workflow that was mentioned in the paragraph before.

### 3.8.3 Design

Design uses a Kanban board with six columns. In Figure 12, can exist:

- Design stories: pieces of work that must be done to achieve a certain Epic;
- Design bugs: problems found and reported that should be fixed.

Design’s Kanban columns are – “To Do”; “In Progress”; “Blocked”; “Internal Review”; “External Review”; “Done”.
When a ticket is picked up by a Team member, it moves from “To Do” into “In Progress”. The “In Progress” column has a maximum of 7 tickets in order to avoid a lot of tickets in progress. Depending on the nature of the work, a ticket can me moved into “Internal Review” or to “External Review”, and if it passes the acceptance criteria it is moved into “Done”.

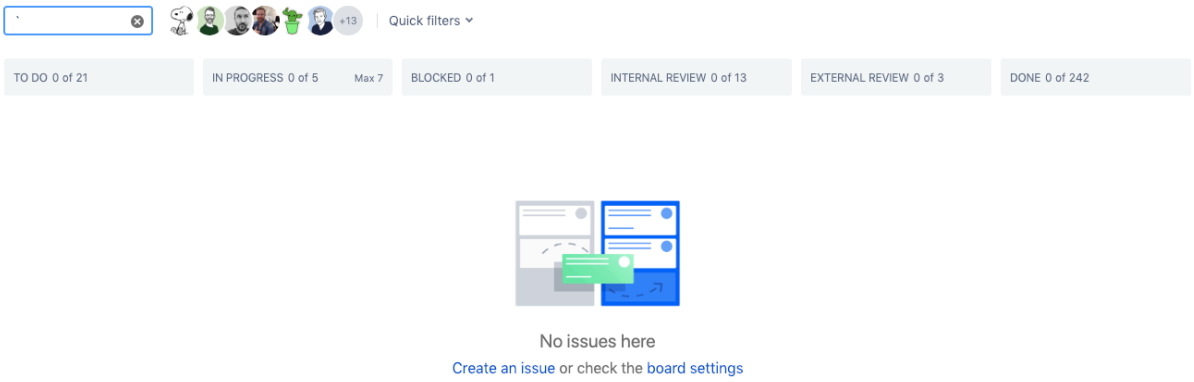


**Figure 12 - Design Kanban Board**

When a ticket is created for Design, the flow follows the User Centred Design. Represented in Figure 13 we can see that.

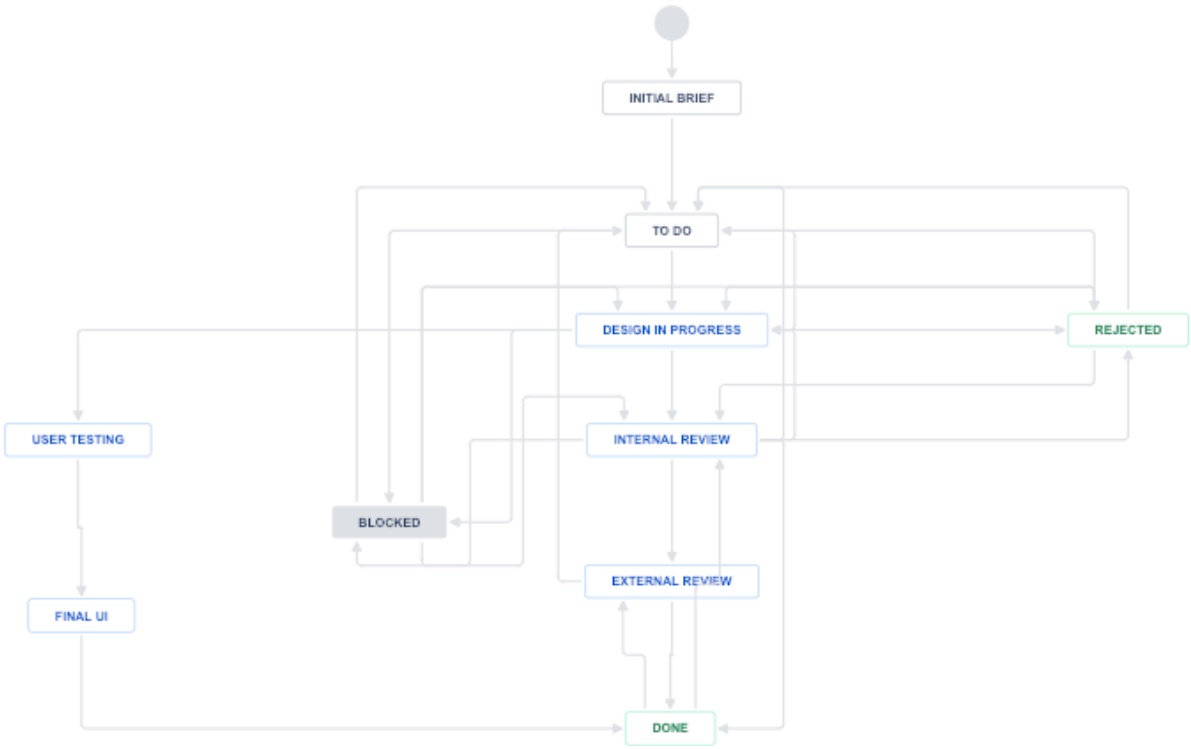


Figure 13 - Design Ticket Workflow

When the ticket is created it gets the “Initial Brief” status. After that, it goes in to the Kanban Backlog, with the status of “To Do”. When a designer moves the ticket in to “In Progress”, the status of the ticket is “Design in Progress”.

From there, a ticket can have the “User Testing” or in “Internal Review” status whilst being in the “Internal Review” column. From here, the ticket can be moved into “External Review” and to “Done” right after, or it can skip the “External Review” if no customer feedback is needed. In any stance a ticket can get the “Rejected” status.

# 4. SOLUTION PROPOSED

Scrum is all about communication, transparency and visibility. It rests on the premises that the Teams are a group of experts that are auto-sufficient and auto-managed. In Living Map, these guidelines were not being followed.

In order to bring the process to what it should be, action was taken to re-implement and re-define scrum. I acted as the Scrum Master and together with the team brought the good practices into place. Our main objective was to make the process clearer, more visible and more transparent.

In this chapter first the problems identification and then the solution proposed will be presented.

## 4.1 Identifying the Problem

### 4.1.1 Scrum Flow

The Sprint would start on a Thursday for both Teams. The ceremonies occurring during the two weeks were the Daily Meetings, Backlog Refinement, Sprint Planning, Sprint Review, and Sprint Retrospective. The Teams would have every ceremony separated with the exception of Sprint Retrospective, meaning that the only person connecting both teams was the Scrum Master (SM)/Technical Architect (TA) at the time.

The Sprint Planning had the duration of an hour where the Team would talk about their velocity in the last Sprints, estimated their new velocity and agreed on their capacity and populated the new Sprint with tickets that are present on the Product Backlog. For this meeting, the Product Owner would be present and help to decide the tickets that go into the next Sprint, based on their importance and urgency. It happened frequently that refinement and estimation occurred in this ceremony, which should not happen. Also, this ceremony was time-boxed, creating a very stressful environment, because there was no space to discuss any implementation of a ticket and the SM/TA would not allow discussion on it either, imposing his own implementation.

After Sprint Planning, the Team met for Retrospective where they discussed what went wrong, what went well and general performance. The Sprint Retrospective is a closing ceremony where only the SM and the Teams attend. Although the ceremonies were secret, the points made were public for everyone to see. It was common to have bullets like:

- We are moving away from Scrum principles;
- Sprint process seems broken;
- Team split communication;
- Retrospective actions are not done;
- We should not assume in “Backlog Refinement”;
- Taking extra work into Sprints.

The Daily Meetings were held every day for fifteen minutes, where the Team answers the three standard questions. It was usual for the daily meetings to drag for a long time, and Team members would complain about the amount of time spent discussing different themes.

The Backlog Refinement was held once every week, for one hour and had the purpose of refining tickets. In the company, Account Managers and Business Developers create tickets for a certain project. Since they are not software developers, when creating these tickets, there is not enough information or technical details on them. The technical architect would go through them and add all the relevant information and how the work should be done. During this ceremony, the TA presents the tickets to the Team and after that, the team proceeds with estimation. The Team estimates in Story Points by using the Planning Poker. Since the Team had no input on how the ticket should be implemented or of what the bigger picture of the project was, estimations would frequently fall short. During the Sprint, implementation problems, dependencies and blockers would appear that were not thought about in the Backlog Refinement, creating a delay in work.

On the last day of Sprint, the Team had the Daily Meeting as usual, and after that, the Sprint Review. Here, all the Teams (Design, GIS and Developers) presented the work they had been working on for the last two weeks. This ceremony lasted for forty-five minutes and was led by the Scrum Master. The Teams expressed their discontent with the non-appearance of directors and of investors in this ceremony.

After Sprint Review, the Team has Sprint Planning, beginning the cycle again.

The 1:1 meeting was supposed to be held whenever a Team member asked for it and was a chance for the Team member to talk with the Scrum Master in private, present his doubts and just discuss whatever he seems fit. In this way, the Scrum Master maintains a close relationship with his Team and has a more general overview of their happiness and satisfaction. These 1:1's were not happening frequently enough and when they did happen, the Team members did not feel at ease with their Scrum Master and had expressed that they felt like their worries and opinions were not listened to.

The Scrum Flow respected the standard ceremonies but totally forgot their purpose. The Scrum Master was imposing his own views on how every work should be implemented, the Teams did not have an active opinion on the work they were doing, and they were discontent on how the process was treating them.

### 4.1.2 Scrum Ceremonies

The Backlog Refinement was one of the roots of the lack of communication and transparency. The process of creating and refining tickets was trusted to one single person, who was both the Scrum Master and Technical Architect. Having had two major Job roles in the company the resulting Scrum performance was poor. The process was as follows:

- Tickets are created by the PO or by a business developer and sent to the TA;
- TA refines how these tickets should be implemented;
- In “Backlog Refinement”, the tickets are introduced to the Teams;
- The Teams would estimate the tickets;
- The tickets created would be a candidate for next sprint.

The problem with this process is the lack of input the Team - the group of experts - has into the ticket creation and implementation. Since they did not have any thoughts on how the tickets should be implemented, their estimation did not take into account the problems that might occur, dependencies, or amount of effort.

This process flow created an environment where the Teams were not happy with what they were implementing, where Sprints were not being completed, tickets were not well refined and took longer than expected to complete.

Furthermore, the only ceremony where the Teams were together was in Sprint Retrospective. There was an alienation between the Teams. They were not aware of what the other Team was working, what was already implemented or what problems might be occurring with a Project. In some occasions, the Teams were planning to implement the same features in the same Sprint. The work between the Teams was not transparent or visible for everyone to check.

### 4.1.3 Ticket Flow

In JIRA the ticket flow can be seen in Figure 11 - Software Development Ticket Workflow. The PO or the Business Developer creates a ticket with a "User Story", under a Project, and relates it to an Epic. This ticket created would start the workflow on the "Requested" column.

The TA, when informed, would take the new ticket and move it to "Refinement". Here, the ticket would be refined, by being populated by the "Acceptance Criteria". After being refined, it would be moved to the "Pre-flight" check, where it would remain until the "Backlog Refinement" ceremony took place, and the ticket was estimated. After being estimated, the ticket would move into the "Backlog" where it would remain until taken into the Sprint.
Both of the Teams used the same Product Backlog but had no idea which tickets would be given to which Team. The only person who knew this was the SM/TA.
The first time a Team would see a ticket was either on "Backlog Refinement" on in "Sprint Planning", not giving enough time for the Team to think about the work that needed to be done.

## 4.2 Solution Proposed

### 4.2.1 Scrum Flow

The first action taken was to gather the Development Team in a meeting called "Re-take Scrum" and teach them Scrum. In this meeting, the main goal was to evaluate their Scrum knowledge and explain the good practices versus the bad practices that were being executed. Since the company was growing and had a lot of new additions, it was necessary to explain what was happening wrong, what should be improved and why the improvements were needed.
The second goal was for the Team to feel like their input was valued and taken into consideration. In order to achieve that, multiple themes were brought to discussion - team composition, ceremonies to have, length and frequency of ceremonies, the definition of done and definition of ready. Lastly, to improve estimation, it was defined how much was one Story Point.
The “Re-Take Scrum” had the following agenda:

1. Back to Basics – What is Scrum?
    a. Quiz (Annex I)
    b. Questions and Answers
2. Deciding as a Team
    a. Re-Creating two teams – who would be in Team A and Team B?
    b. Ceremonies
        i. Which Ceremonies should we adapt?
        ii. How often do we need Scrum Ceremonies?
        iii. Are these Ceremonies with both Teams?

c. Definition of Ready;
d. Definition of Done;

How much is a Story Point?

### 4.2.2 Scrum Ceremonies

One of the biggest changes that had to occur was the process of refining and creating tickets. There was an alienation between the Team and the work that needed to be done. The communication was at fault, and most of the Team members had no idea about what was needed to implement, or the project's deadlines. Besides that, it was hard for the Team to have a bigger view of how something had to be done, what integrations should they be concerned with, because they did not have enough information about the overall project.

In order to change this, the process had a big shift. The Teams would be entirely responsible to refine their tickets, so they could meet the "Definition of Ready", decide on how they should be implemented, and what dependencies exist between work. By doing so, it was expected an improvement of Sprint completion, estimation and general awareness of work.

Every Team would start with one Backlog Refinement and one Ticketing per week (with a total of two ceremonies of each per Sprint) where they had to create enough tickets that met the "Definition of Ready" and were candidates for the next Sprint.

The "Backlog Refinement" was held together, but each Team would refine specific features and in the last ten minutes, there would be a brief, where the tickets would be explained to increase transparency. Ticketing and Sprint Planning would be held separately.

The tickets were created using post-its in "Backlog Refinement", delivered to the Scrum Master that would later create them in Jira, labelled them and put them through the ticket workflow to the next ceremony - "Ticketing".

In order to make Daily Scrums shorter and precise, it was introduced an object that would serve as intermediator, meaning that only the person that was holding the object could speak.

### 4.2.3 Ticket Flow

There are two Teams – Team A and Team B – but the Refinement Board and the Product Backlog was only one. In order to have a distinction between the two Team’s tickets, I started labelling the Tickets with #Team_A and #Team_B. Then, created a filter board called “Refinement A” and “Refinement B”, where only tickets with the label #Team_A or #Team_B

would appear in the “Refinement” column, as seen in Figure 18. By doing so, the Teams knew in advanced which tickets they would have to refine during “Backlog Refinement”.

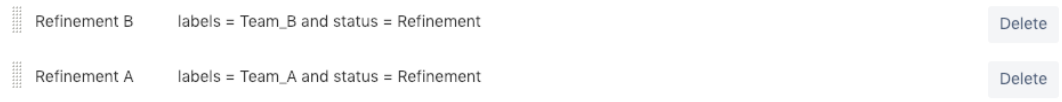


Figure 14 - Refinement Label

After the tickets were refined, they would move to the “Pre-flight Check” column, where a similar filter was applied. The tickets that were created on the “Backlog Refinement” were also labelled and put in the “Pre-flight Check” skipping two columns (Figure 15).


Figure 15 - Ticketing Label

After being moved to the “Backlog” column, they had the label for their Team. In order to make it easier for the PO to prioritise the “Sprint Backlog” of each Team, I labelled the tickets that were candidate for next Sprint with Sprint_I, Sprint_II, Sprint_III,… and created a board filter named “Team_A”, “Team_B”, where it combined the label of the Team with the next Sprint making it simpler for the PO and the Teams to access information (Figure 16).


Figure 16 - Backlog Label

## 4.3 Summary

In summary, the solutions proposed for the problems listed are:

1. *Sprint Completion* – by re-implementing Scrum, it is expected that the good practices lead to a higher Sprint Completion. The goal asked by the CTO was predictability of 80%-90%.
2. *Ceremonies*:
    a. Daily Scrum were very long and not helpful – it will be implemented a rule, where only a person holding a “pretend microphone” has the opportunity to speak. In case of a blockage, the person must specify to have a post-scrum.

b. Ticketing was time-boxed with no opportunity for discussion – by introducing a new meeting “Backlog Refinement” where the teams refine their own tickets, this behavior is expected to cease to exist.
c. Sprint Planning involves ticket refinement and estimation - by introducing a new meeting “Backlog Refinement” where the teams refine their own tickets, this behavior is expected to cease to exist.

3. *Ticket Flow*:
    a. A ticket should never stay in “Blocked” for more than 24 hours – Scrum Master will be in charge to unblock the ticket.
    b. A ticket should never be added in to the Sprint without discussion with the Team – Scrum Master to consult with the Teams and with the Product Owner before any change is made.
    c. A ticket should never be added in to the Sprint without an estimation.
4. *Backlog Visibility* - implementation of labels and filters to make browsing through Jira easier.

# 5. RESULTS

This chapter is divided in six different parts.
In the first part, the initial arrangements and metrics (performance indicators) are explained. Secondly, the conclusion of the "Re-take Scrum" meeting is presented. Thirdly, past data from three sprints prior to the "Scrum re-take" is analysed, followed by part 4, where the data from 6 iterations after "Scrum re-take" is presented. All the data collected is attached on Appendix I.
Following that, part five compares the Team's combined results of the changes seen with the performance indicators, and to end, in chapter sixth the new Scrum Flow implemented is explained.

## 5.1 Performance Indicators

Every Sprint the following data was collected:

- Points Planned – the amount of Story Points each Team compromised to deliver;
- Points Achieved – the amount of Story Points each Team actually delivered;
- % Achieved – Points Achieved divided by Points Planned;
- Points per Day – the amount of Points Achieved divided by the number of Days in the Sprint;
- Points per Person per Day – the amount of Points Achieved divided by the number of Person Days in the sprint.

The Person Days were added manually through data retrieved from another excel spreadsheet represented in Figure 17. Here the following data is presented:

- Hours per day – a day would be represented by 5 hours;
- Days in Sprint – working days within a Sprint;
- Hours / Points – the equivalent to Points/Person/Day;
- Days Absent – number of days a certain member would be absent during the Sprint;
- % Availability – the percentage of a team member's time to work on tickets (e.g. a team leader spends more time helping others completing their tickets than working on his tickets, so his availability will be less than 100%);
- Person Days – $(\frac{Days\ in\ Sprint - Absent\ Days}{100}) * \%Availability$;
- Team Commitment – the capacity the Team estimated and agreed upon;

- Points in Jira – amount of times to be represent in Jira. This is the addition of the Team commitment and the Carry Over;
- Carry Over – the different between the original estimation and the remaining estimate of work on a ticket.

The Carry-Over was necessary to be implemented due to the existence of tickets that were not completed on their current Sprint but had to be continued on the next Sprint. For example, the ticket SA-125 had an original estimate of 3 story points, there was still a point left to do. So, the ticket would carry over with its normal estimate, but only valuing 2 points. The full spreadsheet can be found in Appendix II.

| | |
|---|---|
| HoursPerDay | 5 |
| DaysInSprint | 5 |
| Hours:Points | 0.28 |

| | Days Absent | % Availability | PersonDays | Points/Person |
|---|---|---|---|---|
| Member 1 | 1 | 70 | 2.8 | 3.92 |
| Member 2 | 0.5 | 70 | 3.15 | 4.41 |
| Member 3 | 0.5 | 70 | 3.15 | 4.41 |
| Member 4 | 2 | 50 | 1.5 | 2.1 |
| Member 5 | 0.5 | 55 | 2.475 | 3.465 |
| Member 6 | 1.5 | 65 | 2.275 | 3.185 |
| Member 7 | 3.5 | 65 | 0.975 | 1.365 |

| Total person days | Story Points | Points/Person/day | Team Commitment |
|---|---|---|---|
| 16.325 | 22.855 | 1.4 | 20 |
| | | | Points In Jira |
| | | | 24.5 |

| Carry Over | | | |
|---|---|---|---|
| Ticket | Original | Remaining | Carry Over |
| sa-125 | 3 | 1 | 2 |
| vm-241 | 3 | 0.5 | 2.5 |
| dev-1347 | 8 | 8 | 0 |
| | | | 0 |
| | | | 0 |
| | | | 0 |
| | | | 4.5 |

Figure 17 - Team Capacity Calculator

## 5.2 Re-take Scrum

After this session, the results were as follows:

- Team Composition – two teams were formed, with different focuses. Team A would be focused on Research and Development, having 7 members, while Team B would be client facing projects, having 6 members;
- Sprint Length – the sprint would remain with two-week iterations;
- Ceremonies
    - Daily Stand Up would be held with the presence of both teams in order to improve communication and transparency;
    - Backlog Refinement would be held with the two teams, but they would refine different tickets, and at the end of the ceremony the tickets would be presented to the other team;
    - A ceremony for Ticketing was created where estimation would take place;
    - Sprint Review remained unchanged;
    - Sprint Retrospective remained unchanged;
- Definition of "Ready"
    - Story must be written as a user story;

- Dependencies identified and added to the ticket;
- "Acceptance Criteria" must exist and be understood by the team;
- Story has been estimated by the team;
- UX sketches exist, where appropriate, and are understood by the team;
- Performance criteria exist, where appropriate, and are understood by the team;
- Test script or test steps (Gherkin) exist;
- Deployment plan is in place;
- The team understands how to demo the feature.

- Definition of "Done"
  - Passed "Code Review";
  - Code is deployed to a test environment;
  - Feature is tested against "Acceptance Criteria";
  - Feature passes smoke test;
  - Feature passes regression testing;
  - Feature is documented;
  - Versioning and release notes;
  - Confluence documentation as required;
  - Feature approved by UX designer;
  - Feature approved by Product Owner;
  - Code is merged, and corresponding branches deleted.

## 5.3 Past Results

### 5.3.1 Sprint A

Table 3 - Performance Indicators - Sprint A

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint A | 45 | 35.5 | 78.89% | 9 | 3.9 | 44.8 | 0.79 |
| B | Sprint A | 62.5 | 52.5 | 84.00% | 9 | 5.8 | 26.6 | 1.97 |
| Joined | Sprint A | 107.5 | 88 | 81.86% | 9 | 9.8 | 71.4 | 1.23 |

In Sprint A, Team A planned to achieve 45 story points, completing 35.5 story points, reaching a 78.8% of Sprint completion with 0.79 points per person per day. Team B planned for 62.5 story points, completing 52.5 story points and reaching an 84% of Sprint Completion with 1.97 story points per person per day. In total, the Teams planned for 107.5 story points, completing 88 story points and reaching a total of 81.8% sprint completion with 1.23 story points per person per day (Table 3).

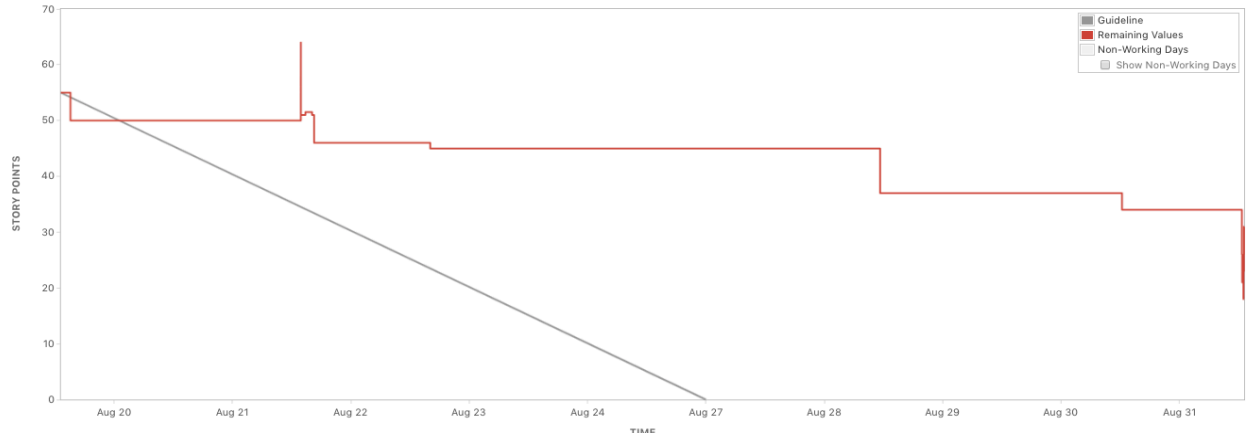


Figure 18 - Burn Down Chart - Team A - Sprint A

Team A accounted for a Sprint with 9 days, but only selected 7 days in Jira, as we can see in Figure 18. There was a peak at 21st August, meaning that the Sprint Backlog was altered by adding and removing tickets. The cadence of the workflow is slow, and the Sprint did not get finished.

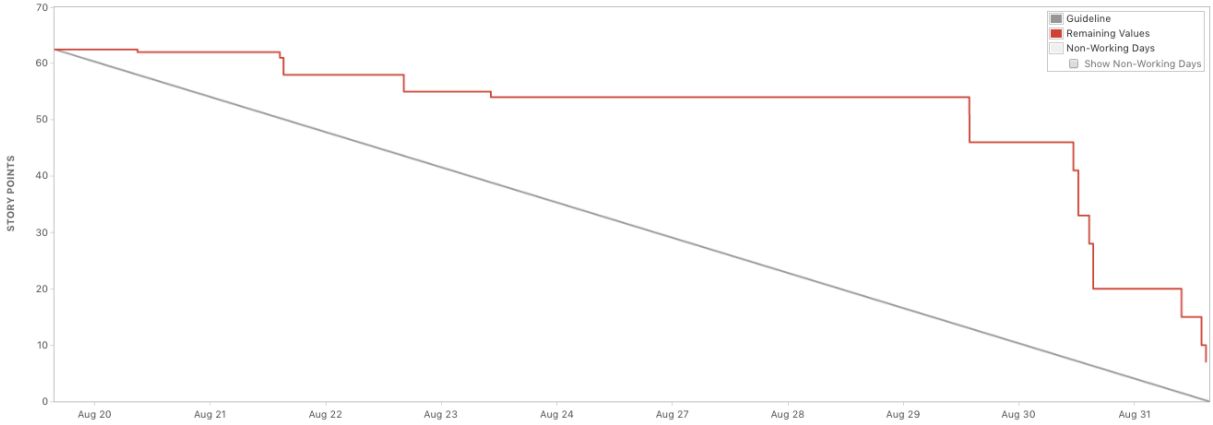


Figure 19 - Burn Down Chart - Team B - Sprint A

Team's B burn down chart is comparatively better to Team's A burn down, however, the tickets are majorly moved in to "Done" on the last two days of Sprint, as we see in Figure 19.

### 5.3.2 Sprint B

Table 4 - Performance Indicators - Sprint B

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint B | 46 | 31 | 67.39% | 8.5 | 3.6 | 23.45 | 1.32 |
| B | Sprint B | 46 | 20.5 | 44.57% | 8.5 | 2.4 | 23.37 | 0.88 |
| Joined | Sprint B | 92 | 51.5 | 55.98% | 8.5 | 6.1 | 46.82 | 1.10 |

In Sprint B, Team A planned to achieve 46 story points, completing 31 story points, reaching a 67.39% of Sprint completion with 1.32 points per person per day. Team B planned for 46 story points, completing 20.5 story points and reaching a 44.57% of Sprint Completion with 0.88

story points per person per day. In total, the Teams planned for 92 story points, completing 51.5 story points and reaching a total of 55.98% sprint completion with 1.10 story points per person per day (Table 4).

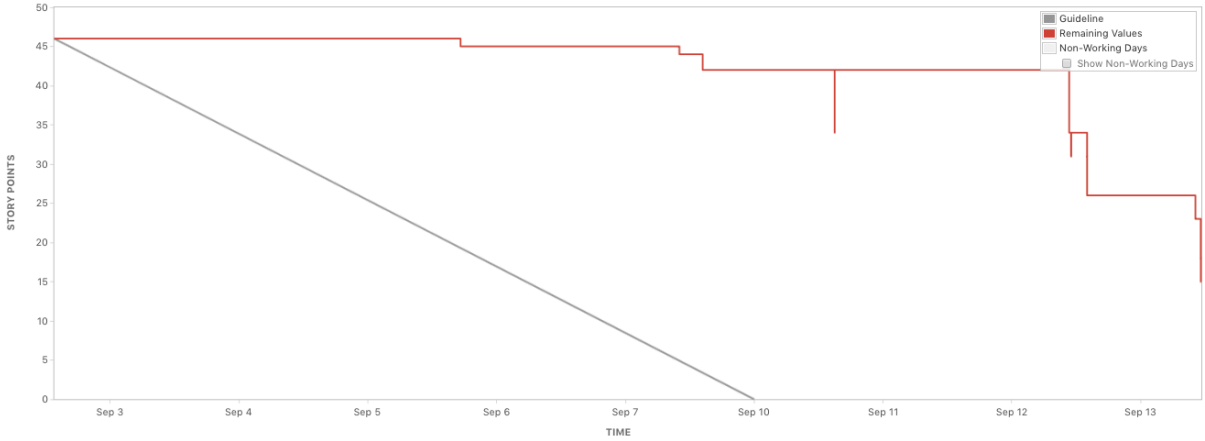


Figure 20 - Burn Down Chart - Team A - Sprint B

In Sprint B, the same problem occurred with the length of the sprint in the burn down chart (Figure 20Figure 20 - Burn Down Chart - Team A - Sprint B). In the 10th of September there was a change in the Sprint Backlog, represented by the inverted peak. The workflow of the tickets was slow, where most of the tickets were pushed into "Done" in the last two days.

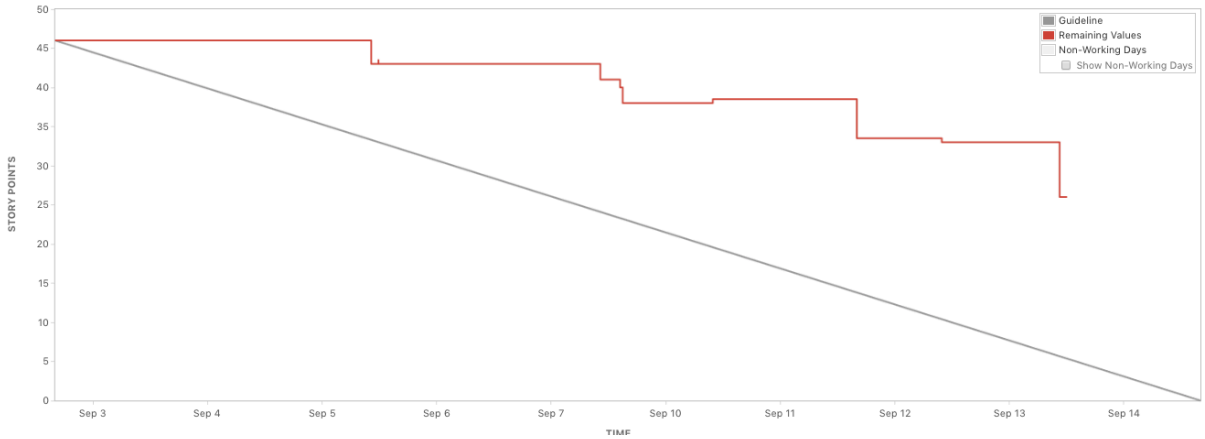


Figure 21 - Burn Down Chart - Team B - Sprint B

The Team's B burn down chart for Sprint B started with a good cadence of tickets, but the velocity was not enough to complete the Sprint (Figure 21).

### 5.3.3 Sprint C

Table 5 - Performance Indicators - Sprint C

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint C | 38.5 | 36 | 93.5% | 9.5 | 3.79 | 20.1 | 1.79 |
| B | Sprint C | 62 | 31 | 50.0% | 9.5 | 3.26 | 21.85 | 1.42 |
| Joined | Sprint C | 100.5 | 67 | 66.7% | 9.5 | 7.05 | 41.95 | 1.60 |

In Sprint C, Team A planned to achieve 38.5 story points, completing 36 story points, reaching a 93.5% of Sprint completion with 1.79 points per person per day. Team B planned for 62 story points, completing 31 story points and reaching a 50% of Sprint Completion with 1.42 story points per person per day. In total, the Teams planned for 100.5 story points, completing 67 story points and reaching a total of 66.7% sprint completion with 1.60 story points per person per day (Table 5).

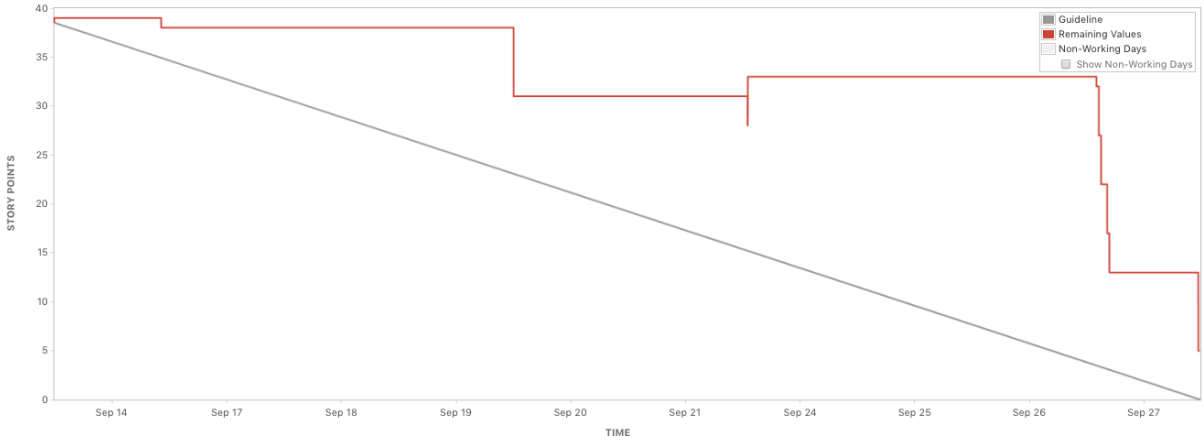


Figure 22 - Burn Down Chart - Team A - Sprint C

Team A's burn down chart improved in Sprint C, but the tendency of have the majority of tickets moved into "Done" in the last days of the Sprint still remains (Figure 22).

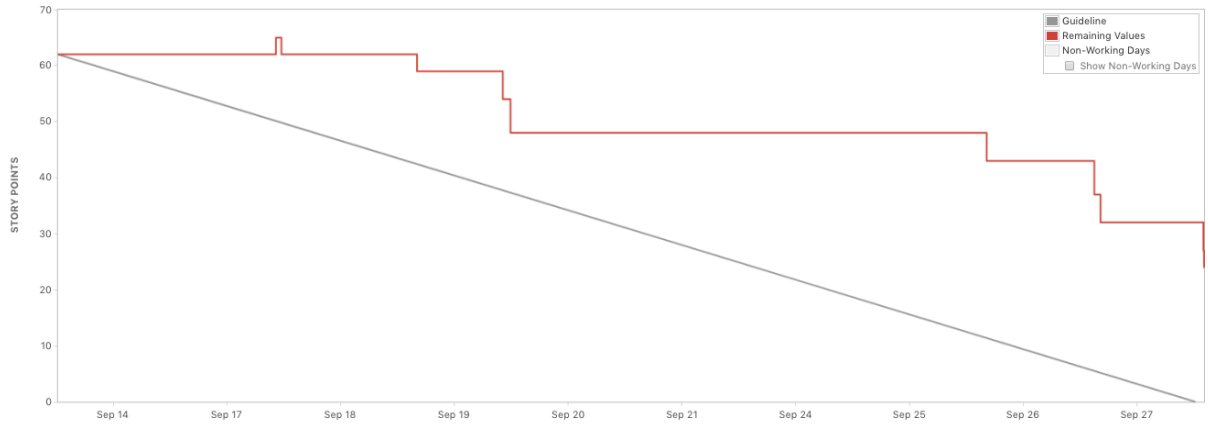


Figure 23 - Burn Down Chart - Team B - Sprint C

In Team B's burn down chart we can see the Sprint was not completed and there were long periods of time where the ticket flow stagnated (Figure 23).

## 5.4 New Results

### 5.4.1 Sprint I

In Sprint I, both teams had already started with the new process. There was separated Backlog Refinement and Ticketing, although some old tickets went in the Sprint since they already were in the product backlog.

Table 6 - Performance Indicators - Sprint I

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint I | 43 | 52.5 | 122.1% | 9 | 5.83 | 28.9 | 1.82 |
| B | Sprint I | 63 | 62 | 98.4% | 9 | 6.89 | 20.45 | 3.03 |
| Joined | Sprint I | 106 | 114.5 | 108.0% | 9 | 12.72 | 49.35 | 2.32 |

Team A planned to deliver 43 story points and achieved 52.5 story points, reaching a 122.1% of Sprint Completion, with an average of 1.82 story points per person per day.
Team B, planned for 63 story points, and achieved 62 story points, with a 98.41% of Sprint Completion and an average of points per person per day of 3.03. In total, they planned for 106 story points, achieved 114.5 story points. Their points per person per day were 2.32 (Table 6).

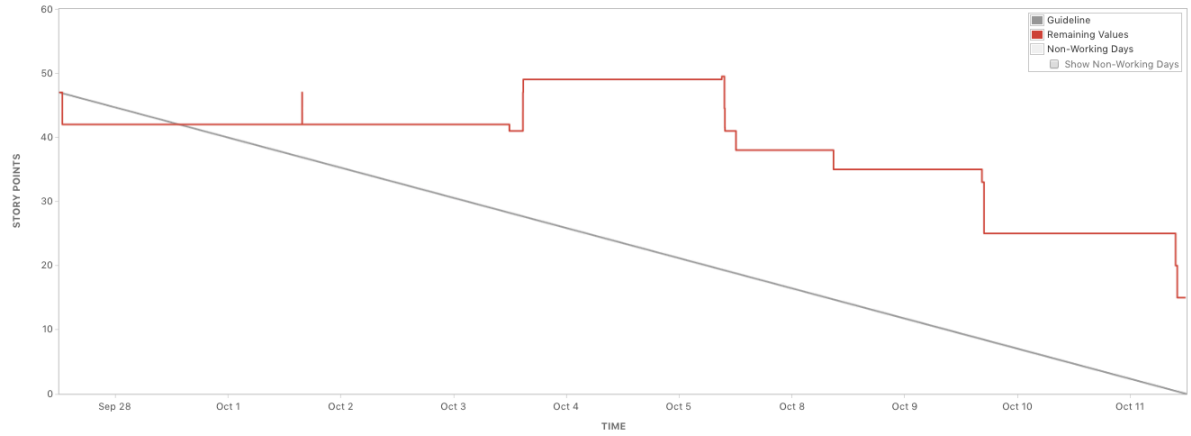


Figure 24 - Burn Down Chart - Team A - Sprint I

In Team A's burn down chart, Figure 24, we can see that right in the beginning, the chart went down the guide line. Team A had changes on their Sprint Backlog, whilst Team B had no changes (Figure 25). The Sprint Backlog changes were necessary because Team Members were complaining about not having enough work to last for two weeks. Even though Sprint Backlog changes should be avoided, they were accepted because the Team was still learning about their Sprint Capacity.

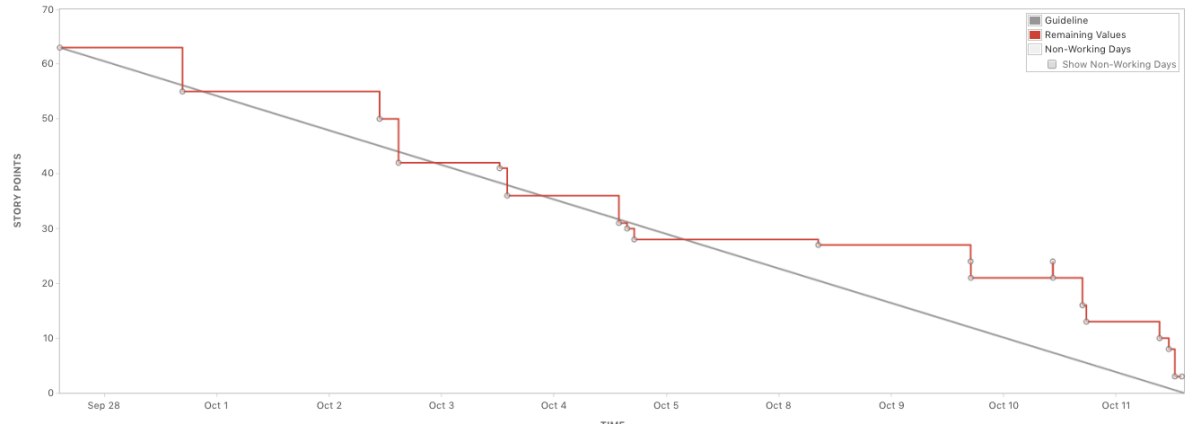


Figure 25 - Burn Down Chart - Team B - Sprint I

Team B's burn down chart almost had a perfect cadence, with tickets being moved to "Done" throughout the all sprint length.

### 5.4.2 Sprint II

Table 7 - Performance Indicators - Sprint II

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Iteration II | 68 | 52 | 76.47% | 9 | 5.8 | 40 | 1.30 |
| B | Iteration II | 54 | 57.5 | 106.48% | 9 | 6.4 | 24.25 | 2.37 |
| Joined | Iteration II | 122 | 109.5 | 89.75% | 9 | 12.2 | 64.25 | 1.70 |

In Sprint II, the ceremonies remained unaltered to Sprint I, and Team A planned to deliver 68 story points and achieved 52 story points, reaching a 76.47% of Sprint Completion, with an average of 1.30 story points per person per day.

Team B, planned for 54 story points, and achieved 57.5 story points, with a 106.48% of Sprint Completion and an average of points per person per day of 2.37.

In total, they planned for 122 story points and achieved 109.5 story points. Their points per person per day were 1.70 per person per day (Table 7).

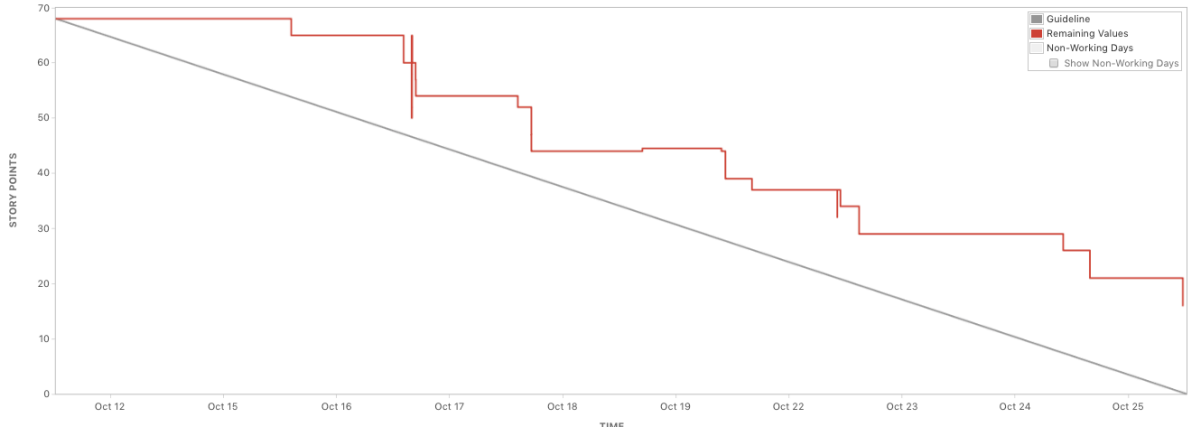


Figure 26 - Burn Down Chart - Team A - Sprint II

In Sprint II, Team A's burn down chart achieved a satisfactory form, having a constant flow on tickets moved in to "Done". It is possible to see that the Sprint Backlog had changed in the beginning of the iteration (Figure 26).

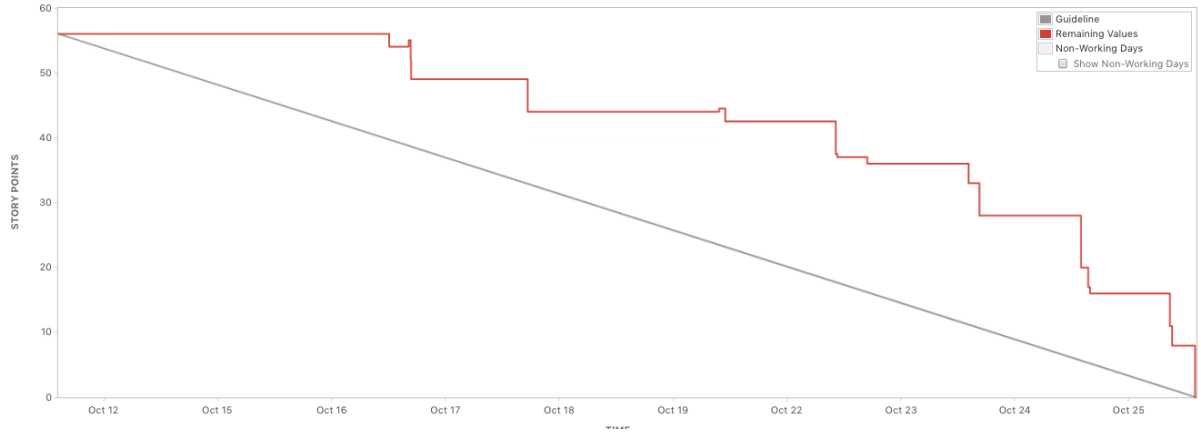


Figure 27 - Burn Down Chart - Team B - Sprint II

Team B achieved a very good burn down chart, leading to Sprint Completion (Figure 27). Ideally, there would be tickets moved in to "Done" earlier in the Sprint.

### 5.4.3 Sprint III

In Sprint III, it was decided that Team A would run into one-week sprints. This decision was made in order to minimize risk. Team A focuses on Research and Development, and their findings of one week were demanding urgent implementation tickets, or new research tickets. The Team would do research and implementation of tickets from Thursday to Monday, and on Tuesday, the tester would test all the changes that were made. On Wednesday, and based on the results of testing, there would be Backlog Refinement and Ticketing in order to prepare to the next sprint starting on Thursday again. A two-week sprint did not allow for a good planning of work because it was too long, and it did not answer to the reactive nature of the team and of the Project it was working on.

Table 8 - Performance Indicators - Sprint III

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint III | 43 | 46.5 | 108.1% | 5 | 9.30 | 24 | 1.94 |
| A | Sprint III | 40 | 19 | 47.5% | 4 | 4.75 | 20 | 0.95 |
| B | Sprint III | 66 | 59 | 89.4% | 9 | 6.56 | 24.7 | 2.39 |
| Joined | Sprint III | 149 | 124.5 | 83.6% | 9 | 13.83 | 68.7 | 1.81 |

Team A planned to deliver 43 story points and achieved 46.5 story points, reaching a 108.14% of Sprint Completion, with an average of 1.94 story points per person per day on their first half.

On the second half, Team A planned for 40 story points, achieving 19 story points, reaching a 47.50% of Sprint Completion and an average of 0.95 story points per person per day.

Team B, planned for 66 story points, and achieved 59 story points, with an 89.39% of Sprint Completion and an average of points per person per day of 2.39.

In total, they planned for 149 story points and achieved 124.5 story points. Their points per person per day were 1.81 (Table 8).

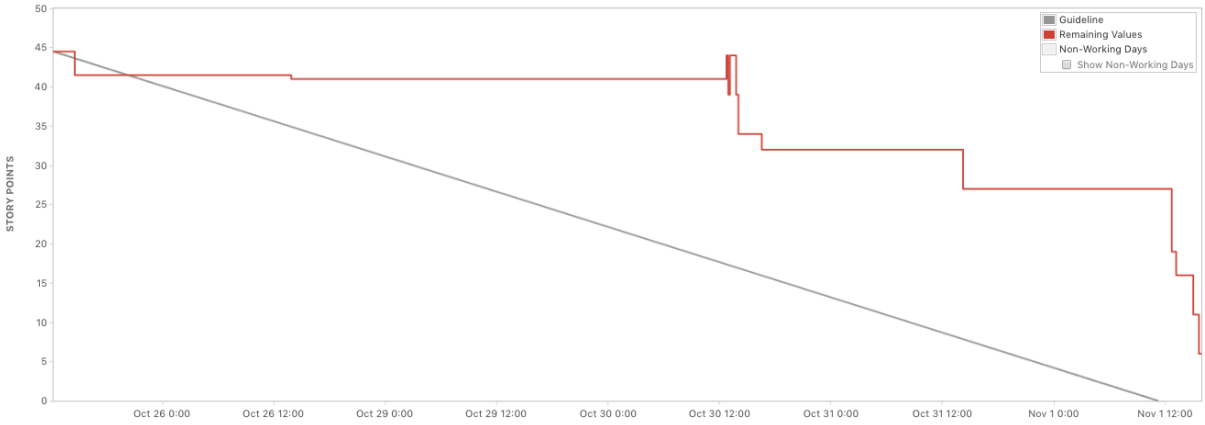


Figure 28 - Burn Down Chart - Team A - Sprint III

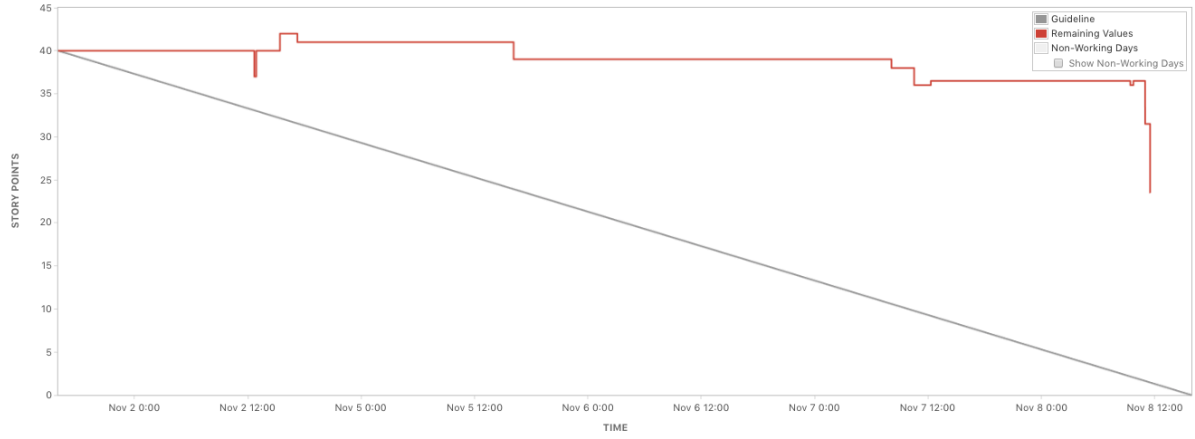


Figure 29 - Burn Down Chart - Team A – Sprint III.2

In the first part of the Sprint (Figure 32), tickets only started moving into “Done” on the third day of the Sprint. After that the cadence was regular and significant. In the second part of the Sprint (Figure 33) the burn down chart barely changes, except on the last day.

By moving in to one-week Sprints, it is possible to see that the flow of the tickets worsen. Tickets only get finished by the end of the iteration.

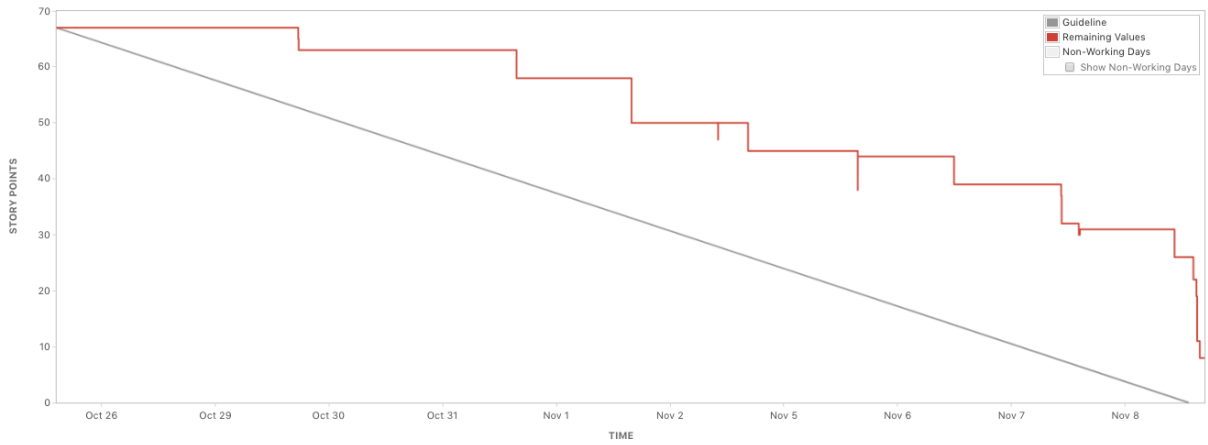


Figure 30 - Burn Down Chart - Team B – Sprint III

Team B's burn down chart was very positive. They almost finished the sprint and had a ticket flow starting in day two, keeping it throughout all the sprint (Figure 30).

### 5.4.4 Sprint IV

In Sprint IV there was a lot of disturbance from the management team to see progress on the Project the Teams were working on. Because of this, the Sprint Scope changed often, the teams were disrupted and there was high pressure to show something tangible to the director.

The pressure to react and plan based on testing in Team A was very stressful and demanded extra work in order to get ready to next Sprint. Only extremely relevant tickets would go in, and every Team member had to understand their importance.

Table 9 - Performance Indicators – Sprint IV

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint IV | 39.5 | 20 | 50.6% | 5 | 4.00 | 24.2 | 0.83 |
| A | Sprint IV | 33 | 28 | 84.8% | 4 | 7.00 | 21.49 | 1.30 |
| B | Sprint IV | 78.5 | 61 | 77.7% | 9 | 6.78 | 33 | 1.85 |
| Joined | Sprint IV | 151 | 109 | 72.2% | 9 | 12.11 | 77.2 | 1.41 |

Team A planned to deliver 39.5 story points and achieved 20 story points, reaching a 50.63% of Sprint Completion, with an average of 0.83 story points per person per day, for the first part of their sprint. In the second part of Sprint IV, Team A planned to deliver 33 story points and achieved 28 story points, reaching a 84.85% of Sprint Completion, with an average of 1.30 story points per person per day. Team B, planned for 78.5 story points, and achieved 61 story points, with a 77.71% of Sprint Completion and an average of points per person per day of 1.85. In total, they planned for 151 story points and achieved 109 story points. Their points per person per day were 1.41 (Table 9).

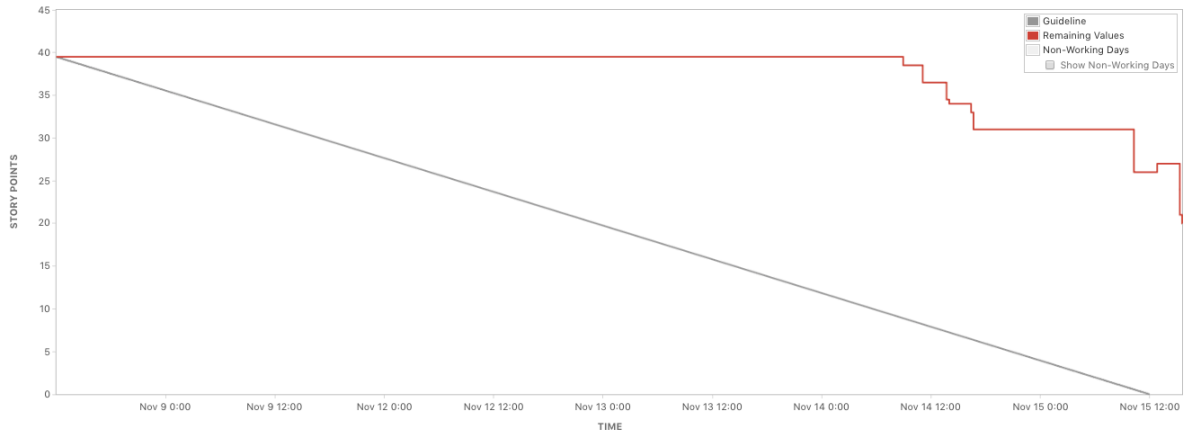


Figure 31 - Burn Down Chart - Team A - Sprint IV

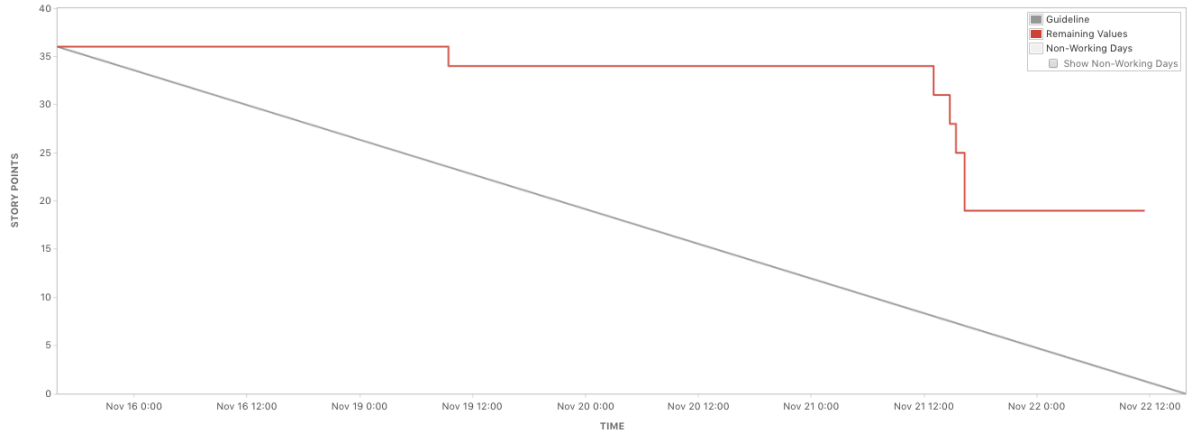


Figure 32 - Burn Down Chart - Team A - Sprint IV.2

The tendency observed in the burn down chart in Sprint III was observed also in Sprint IV. Both charts (Figure 31 and Figure 32) show that the tickets were only moved into “Done” on the last days of the Sprint. Also, both Sprint were not completed.

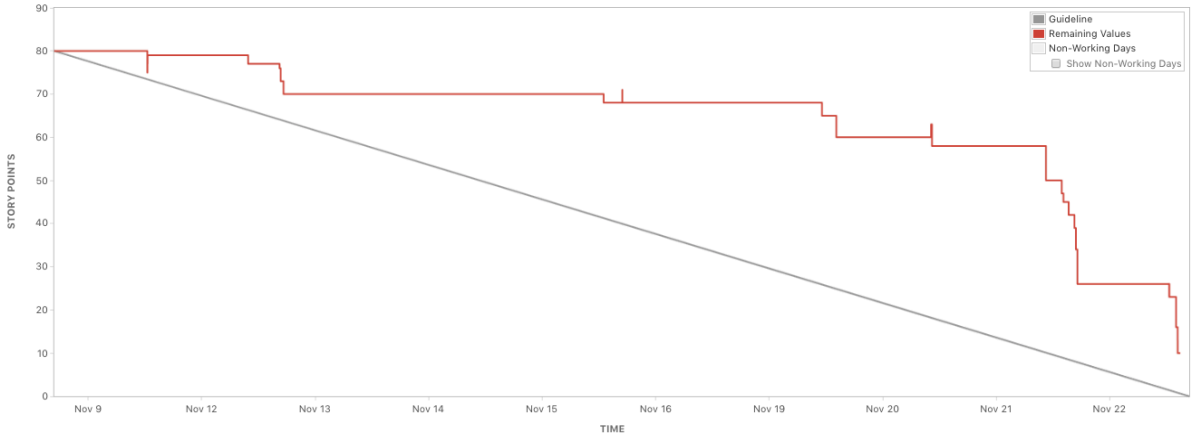


Figure 33 - Burn Down Chart - Team B - Sprint IV

In Sprint IV, Team B's productivity suffered through all the external requests that came from management. During mid Sprint, the Team was asked to do off-sprint work, which interfered with the delivery of the Sprint Goal. In Figure 33 we can see that a big part of the tickets was only finished during the last days of the Sprint.

During Sprint IV, it was introduced one more Backlog Refinement per week for Team B, because of the increasingly capacity the Team was estimated to achieve.

### 5.4.5 Sprint V

**Table 10 - Performance Indicators – Sprint V**

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint V | 41 | 16 | 39.0% | 5 | 3.20 | 23.6 | 0.68 |
| A | Sprint V | 46 | 40 | 87.0% | 4 | 10.00 | 18.9 | 2.12 |
| B | Sprint V | 88.5 | 77.5 | 87.6% | 9 | 8.61 | 33.3 | 2.33 |
| Joined | Sprint V | 175.5 | 133.5 | 76.1% | 9 | 75.80 | 78.99 | 1.69 |

In Sprint V, Team A planned to deliver 41 story points and achieved 16 story points, reaching a 39% of Sprint Completion, with an average of 0.68 story points per person per day, for the first part of their sprint. In their second part of Sprint V, Team A planned to deliver 46 story points and achieved 40 story points, reaching an 87% of Sprint Completion, with an average of 2.12 story points per person per day. Team B, planned for 88.5 story points, and achieved 77.5 story points, with an 87.6% of Sprint Completion and an average of points per person per day of 2.33. In total, they planned for 175.5 story points and achieved 133.5 story points. Their points per person per day were 1.69 (Table 10).

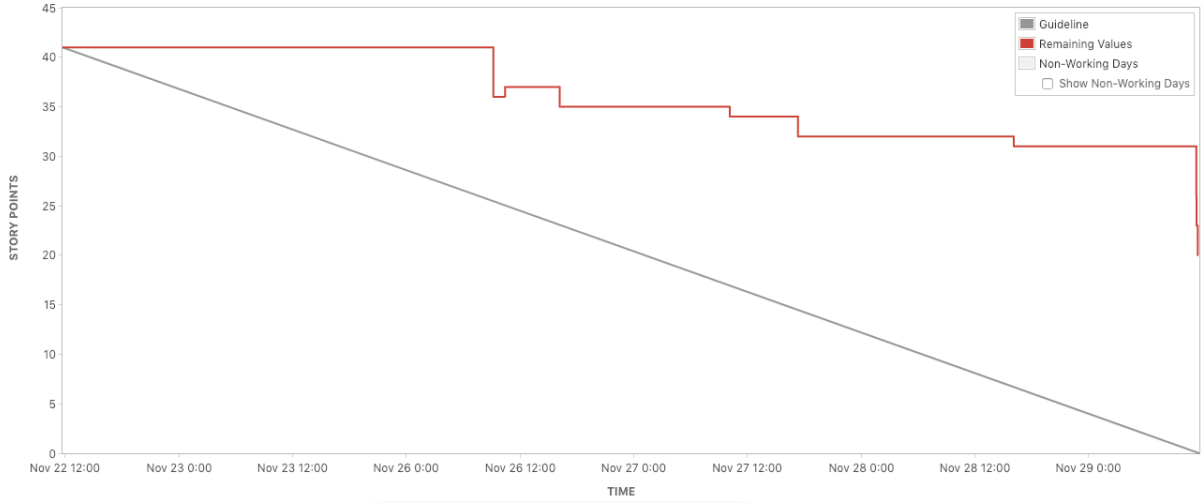


**Figure 34 - Burn Down Chart - Team A - Sprint V**

In the first part of Sprint V, Team A made some progress in terms of their burn down flow. Even though the first ticket moves into "Done" was after 2 days of the Sprint beginning, the tickets kept moving throughout the rest of the iteration (Figure 34).

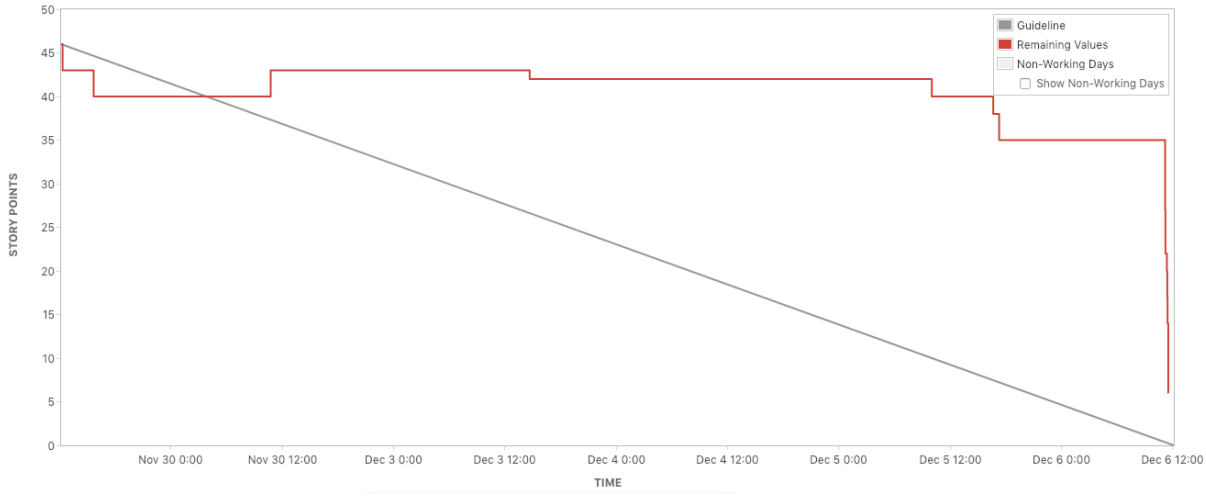


Figure 35 - Burn Down Chart - Team A - Sprint V.2

In the second part of the Sprint though, the flow was not as good as expected, and the team went back to have their tickets finalized through the end of the Sprint (Figure 35).

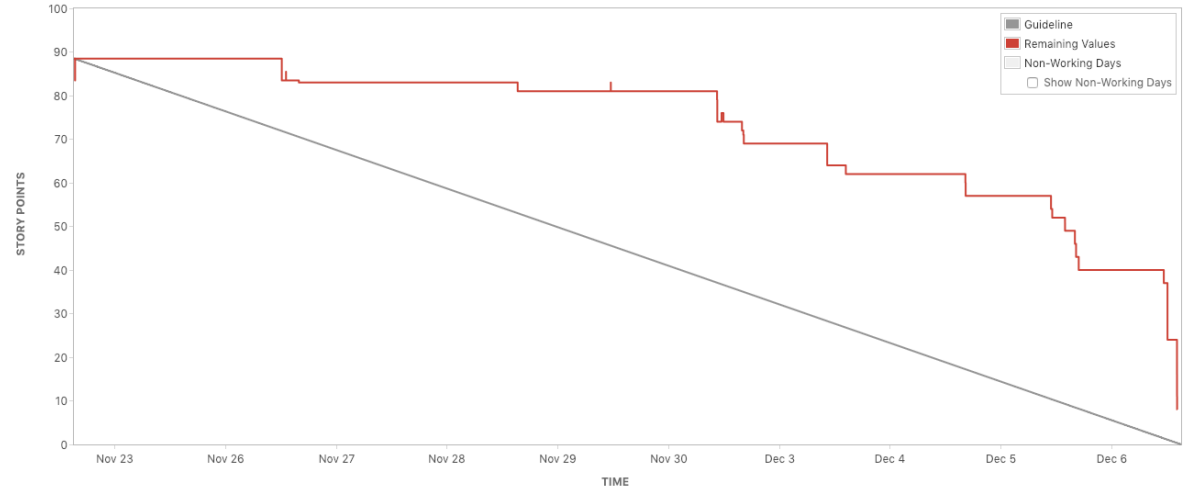


Figure 36 - Burn Down Chart - Team B - Sprint V

In Sprint V, Team B returned to their normal flow. The burn down chart kept decreasing as expected (Figure 36).

### 5.4.6 Sprint VI

Table 11 - Performance Indicators – Sprint VI

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint VII | 34 | 23.5 | 69.1% | 5 | 4.70 | 20 | 1.18 |
| A | Sprint VII | 40 | 40 | 100.0% | 4 | 10.00 | 17 | 2.35 |
| B | Sprint VII | 91 | 91 | 100.0% | 9 | 10.11 | 38.7 | 2.35 |
| Joined | Sprint VII | 165 | 154.5 | 93.6% | 9 | 53.40 | 75.7 | 2.04 |

In Sprint VI, Team A planned to deliver 34 story points and achieved 23.5 story points, reaching a 69.1% of Sprint Completion, with an average of 1.18 story points per person per day, for the

first part of their sprint. In their second part of Sprint VI, Team A planned to deliver 40 story points and achieved 40 story points, reaching an 100% of Sprint Completion, with an average of 2.35 story points per person per day. Team B planned for 91 story points, and achieved 91 story points, with an 100% of Sprint Completion and an average of points per person per day of 2.04. In total, they planned for 165 story points and achieved 154.5 story points. Their points per person per day were 2.04 (Table 11).

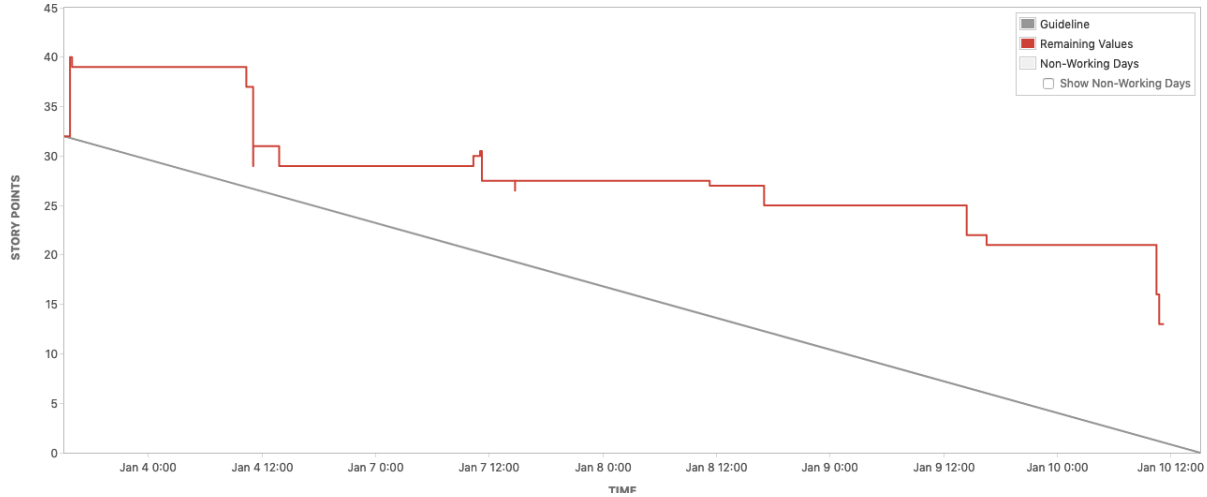


Figure 37 - Burn Down Chart - Team A - Sprint VI

In the first part of Sprint VI, it was needed to add a placeholder ticket to accommodate support in testing. After that, the sprint had a good cadence, with tickets rolling over steadily (Figure 37).

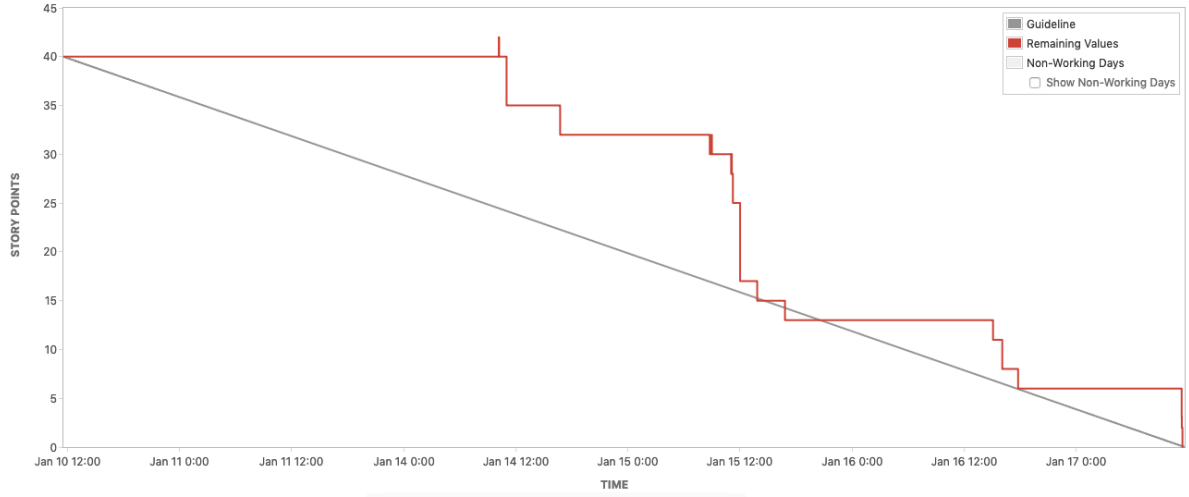


Figure 38 - Burn Down Chart - Team A - Sprint VI.2

In the second part of the Sprint it took some time to move tickets into “Done”. The Team A accomplished sprint completion, and the tickets rolled over throughout the flow with a very good pace, touching the line multiple times (Figure 38).

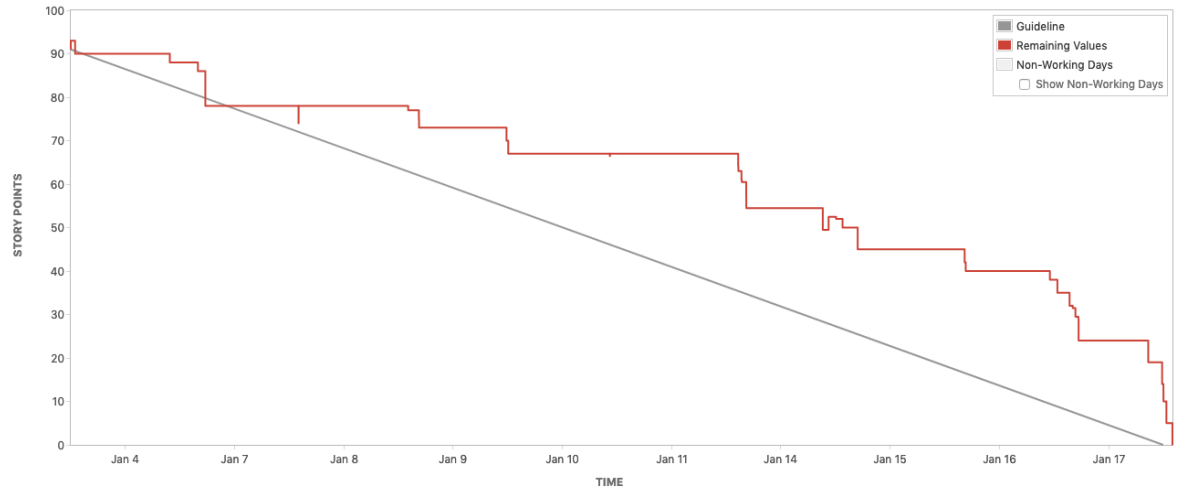


**Figure 39 - Burn Down Chart - Team B - Sprint VI**

In Sprint VI, Team B achieved sprint completion. During the sprint, the tickets were moved to “Done”, although, we can see that a big amount of tickets were moved on the last days (Figure 39). Both teams share a tester, and we can see that from the 9th of January, to the 10th of January, the tickets from Team B were not moved into “Done”, and this happened because the tester was focused on passing Team A’s tickets (Figure 37).

## 5.5 Overall Performance

Using four performance indicators from the Team – points planned, points achieved, points per person per day and percentage achieved – we are able to see the development of the Teams.

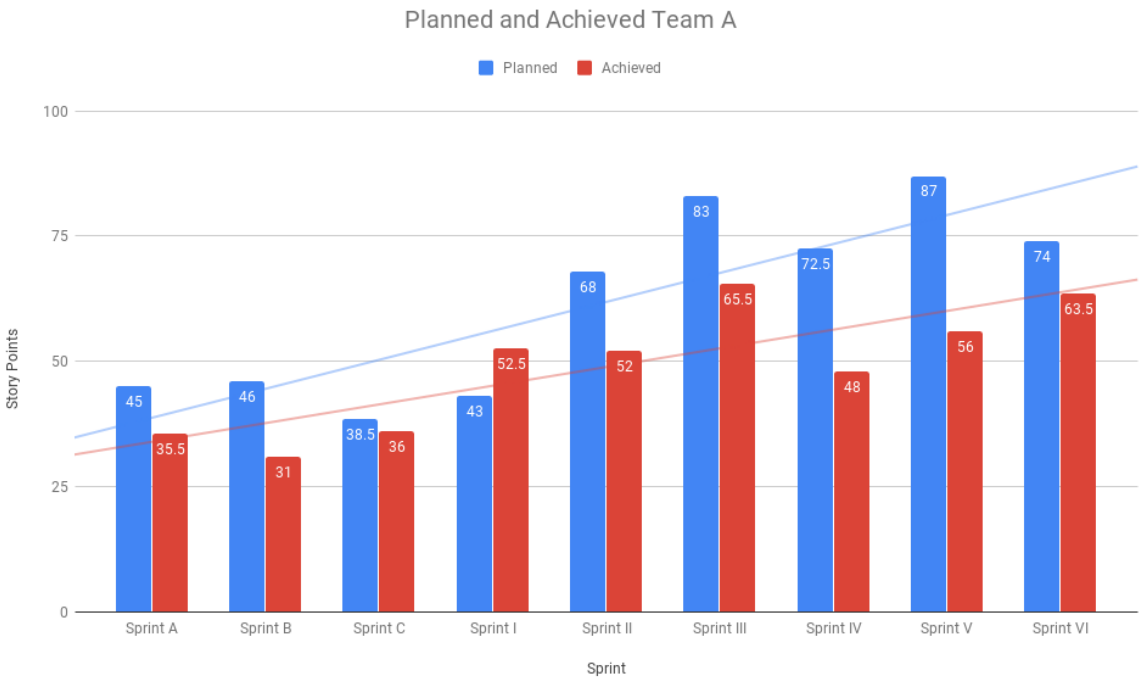


**Figure 40 - Points Planned and Achieved - Team A**

Team A started with planning 45 points in Sprint A, 46 points in Sprint B and 38.5 points in Sprint C to expecting to achieve 43 points in Sprint I, 68 points in Sprint II, 83 points in Sprint III, 72.5 points in Sprint IV and 87 points in Sprint V. We can also see a steady increase on the points achieved. The Team started with 35.5 points in Sprint A, 31 in Sprint B and 36 in Sprint C to deliver 52.5 story points in Sprint I, 52 in Sprint II, 65.5 in Sprint III, 48 in Sprint IV, 56 story points in Sprint V and 74 in Sprint VI (Figure 40).

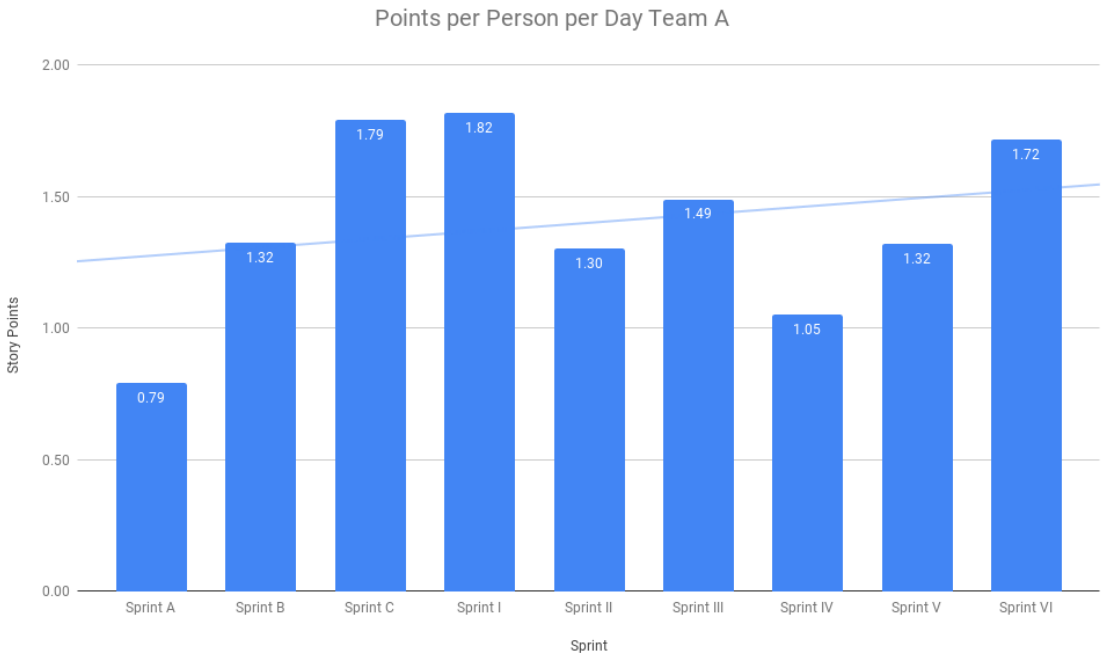


**Figure 41 - Points Person/Day Team A**

Their points per person per day (Figure 41) did not have such a steady growth, going from 0.79 in Sprint A, 1.32 in Sprint B and 1.79 in Sprint C to 1.82 story points in Sprint I, 1.30 in Sprint II, 1.4 in Sprint III, 1.05 in Sprint IV, 1.32 in Sprint V and a bigger growth in Sprint VI, with 1.72 points per person per day.


**Figure 42 - Sprint Goal Achievement Team A**

Regarding the percentage of completion of the Sprint in Figure 42, Team A went from 78.9%, 67.4% and 93.5% in Sprint A, Sprint B and Sprint C to 122.1% in Sprint I, 76.5% in Sprint II, 78.9% in Sprint III, 66.2% in Sprint IV ,64.4% in Sprint V to 85.8% in Sprint VI.

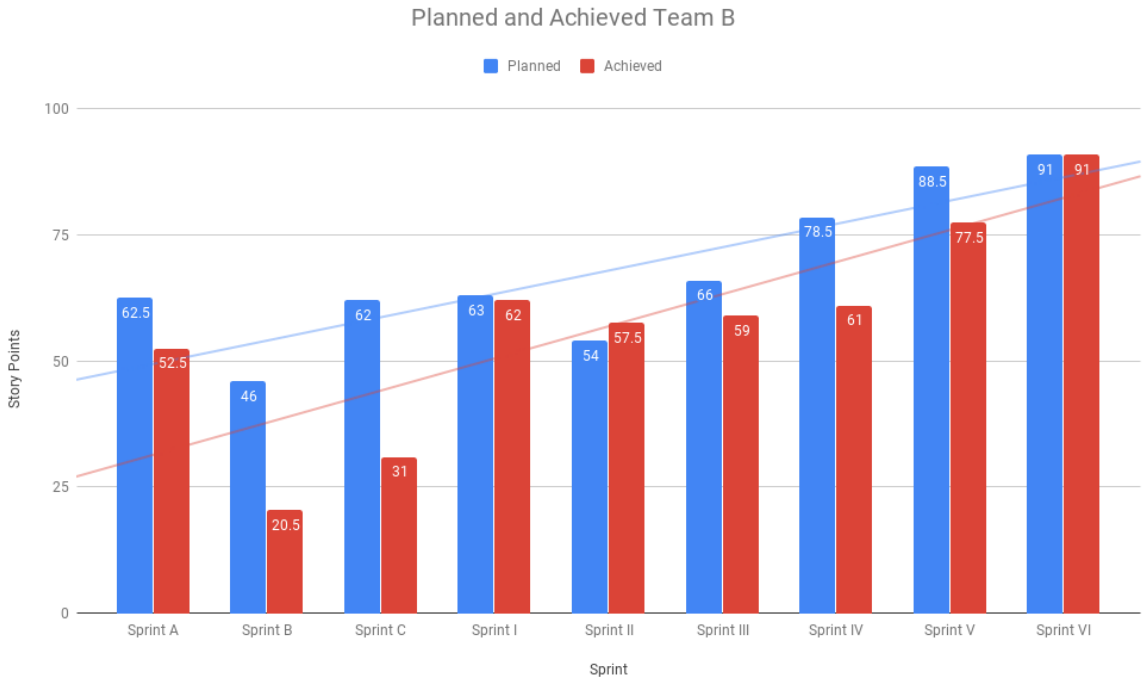


**Figure 43 - Points Planned and Achieved Team B**

Team B started with planning 62.5 points in Sprint A, 46 points in Sprint B and 62 points in Sprint C to expecting to achieve 63 points in Sprint I, 54 points in Sprint II, 66 points in Sprint III, 78.5 points in Sprint IV, 88.5 points in Sprint V and 91 points in Sprint VI. In Figure 43 we can also see a steady increase on the points achieved. The Team started with 52.5 points in Sprint A, 20.5 in Sprint B and 31 in Sprint C to deliver 62 story points in Sprint I, 57.5 in Sprint II, 59 in Sprint III, 61 in Sprint IV,77.5 points in Sprint V and 91 points in Sprint VI.

Figure 44

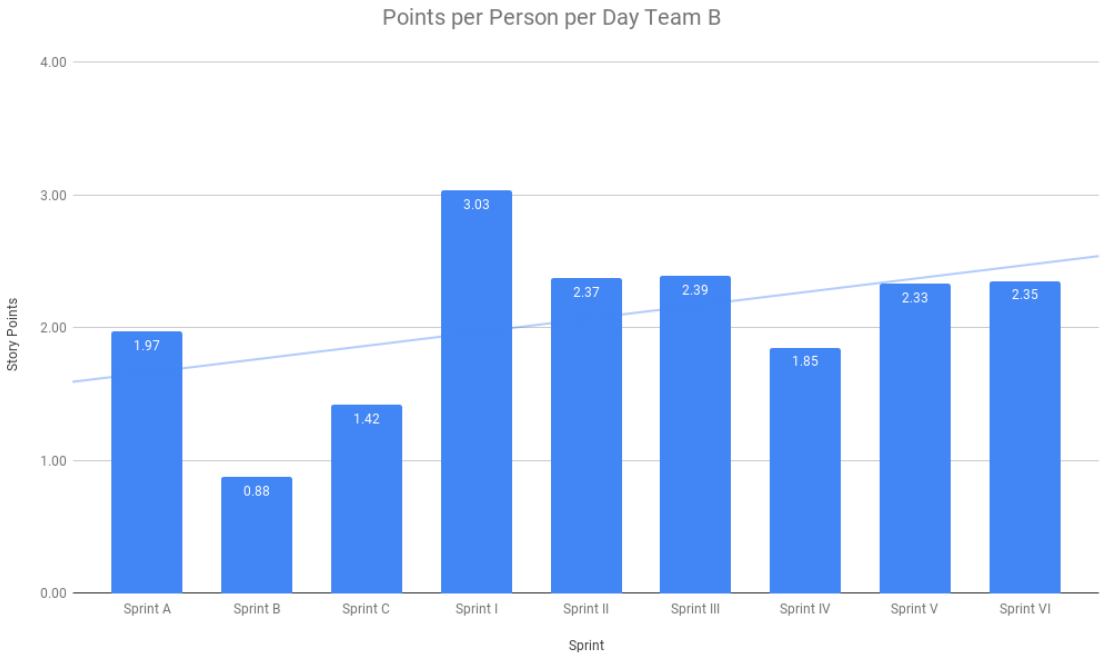


**Figure 44 - Points Person/Day Team B**

Their points per person per day (Figure 44) did not have such a steady growth, having a lot of ups and downs, going from 1.97 in Sprint A, 0.88 in Sprint B and 1.42 in Sprint C to 3.03 story points in Sprint I, 2.37 in Sprint II, 2.39 in Sprint III, 1.85 in Sprint IV, 2.33 in Sprint V and finally 2.35 points in Sprint VI. Sprint IV presents the lowest value achieved due to the management interference in the sprint goal, as it was mentioned in Sprint IV.


**Figure 45 - Sprint Goal Achievement Team B**

In terms of percentage of Sprint Completion in Figure 45, Team B went from 84%, 44.6% and 50% in Sprint A, Sprint B and Sprint C to 98.4% in Sprint I, 106.5% in Sprint II, 89.4% in Sprint III, 77.7% in Sprint IV, 87.6% in Sprint V and 100% in Sprint VI.

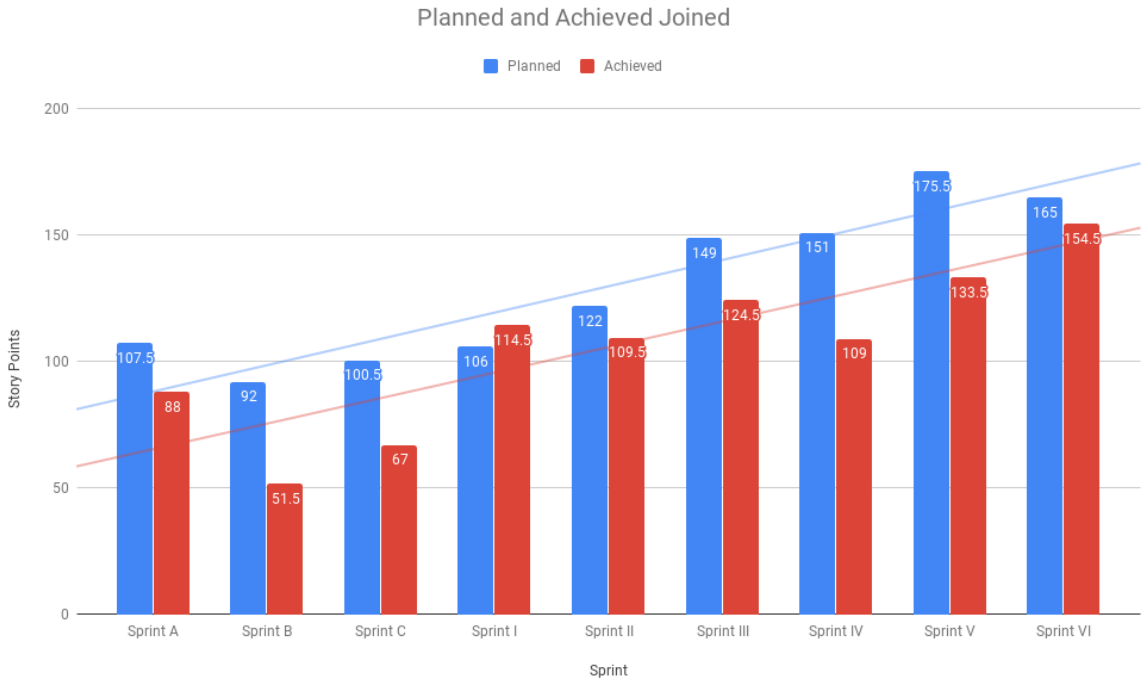


**Figure 46 - Points Planned and Achieved Joined**

Both Teams started with planning 107.5 points in Sprint A, 92 points in Sprint B and 100.5 points in Sprint C to expecting to achieve 106 points in Sprint I, 122 points in Sprint II, 149 points in Sprint III, 151 points in Sprint IV, 175.5 points in Sprint V and 165 points in Sprint VI. In Figure 46, we can also see a steady increase on the points achieved. The Team started with delivering 88 points in Sprint A, 51.5 in Sprint B and 67 in Sprint C to deliver 114.5 story points in Sprint I, 109.5 in Sprint II, 124.5 in Sprint III, 109 in Sprint IV, 133.5 points in Sprint V and 154.5 points in Sprint VI (Figure 46).

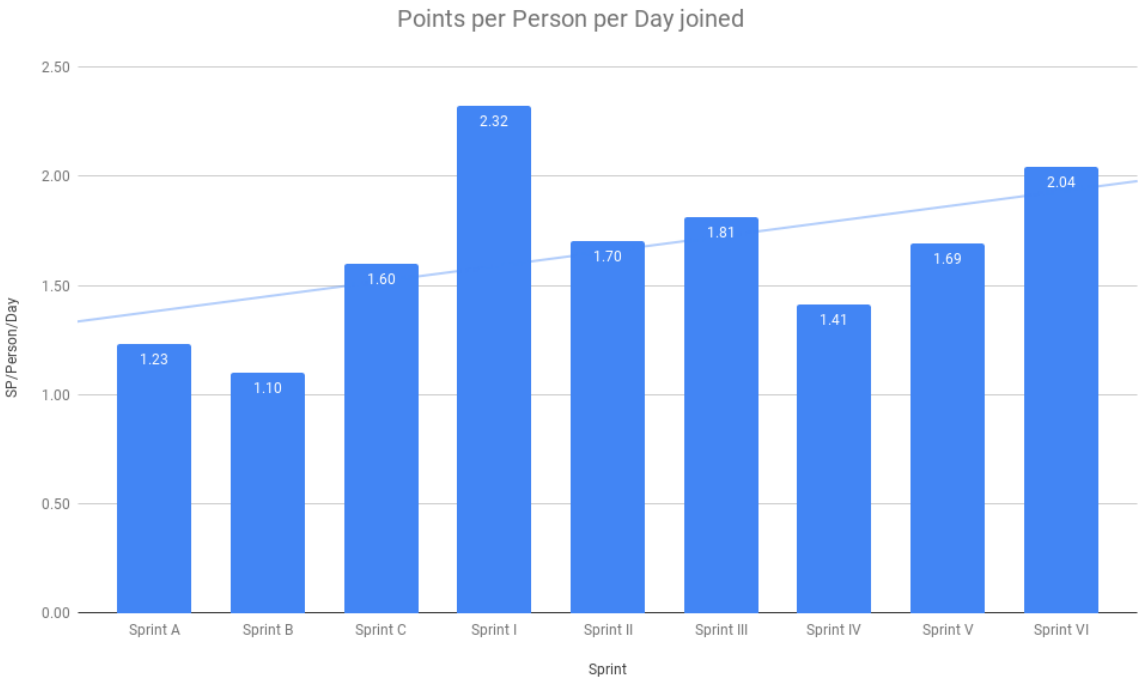


**Figure 47 - Points Person/Day Joined**

Their points per person per day (Figure 47) started from 1.23 in Sprint A, 1.10 in Sprint B and 1.60 in Sprint C to 2.32 story points in Sprint I, 1.70 in Sprint II, 1.81 in Sprint III, 1.41 in Sprint IV, 1.69 in Sprint V and 2.04 points per person per day in Sprint VI.

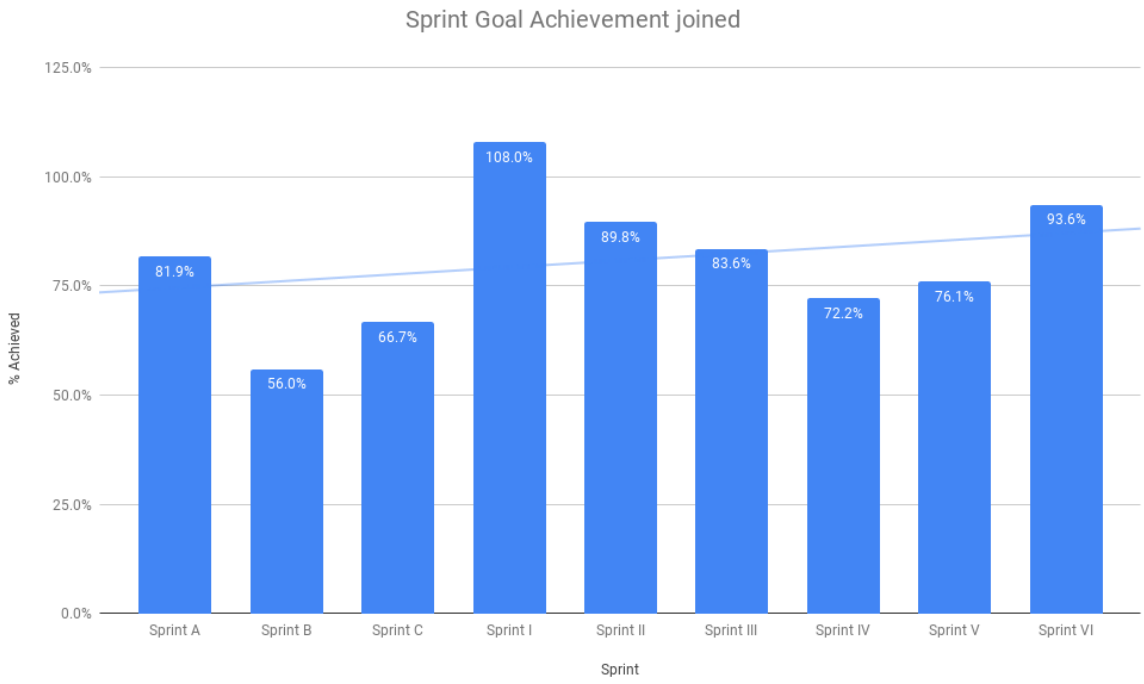


**Figure 48 - Sprint Goal Achievement Joined**

The overall percentage of Sprint completion can be seen in Figure 48, had a slight increase, passing from 81.9% in Sprint A, 56% in Sprint B and 66.7% in Sprint C to 108% in Sprint I, 89.8% in Sprint II, 83.6% in Sprint III, 72.2% in Sprint IV, 76.1% in Sprint V and 93.6% in Sprint VI.

## 5.6 New Scrum Flow

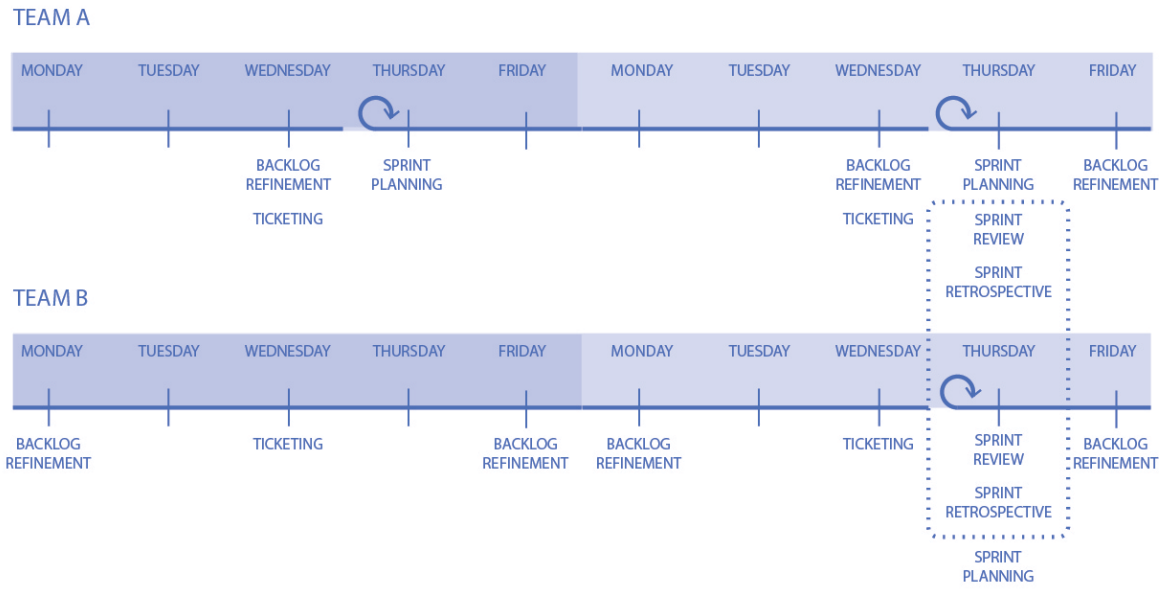


Figure 49 - New Scrum Flow

The Sprint starts on a Thursday and lasts for two weeks in Team B and for one week in Team A. There are some fundamental ceremonies occurring during this two week – Daily Meetings, Backlog Refinement, Ticketing, Sprint Planning, Sprint Review and Sprint Retrospective (Figure 49).

The Sprint Planning has the duration of an hour, where the Team talks about their velocity in the last Sprints, estimates and agrees on their capacity and populates the new Sprint with tickets that are present on the Product Backlog. For this meeting, the Product Owner is present and helps deciding the tickets that go into next Sprint based on their importance and urgency.

After Sprint Planning, the Team meets for Retrospective where they discuss what went wrong, what went well and general performance. From here, some points discussed were:

- Increased transparency;
- Great to have joined Daily meetings;
- Backlog Refinement is great;
- Great to complete Sprints;
- Working together;
- Great Team attitude;
- Great Sprint Planning.

The Daily Meetings are held every day for fifteen minutes, where the Team answers the three standard questions. There is also the right to ask for a Post-Scrum by any member of the team. A Post Scrum is an issue, a blockage or a doubt that a Team member has come across and wants feedback from the Scrum Master or opinion from the team on the best way to solve it. This was implemented in order to avoid off-topic subjects and to allow the Team members to decide if they want to participate in the discussion by staying on the meeting or not participating and leaving. The Daily Meeting, Retrospective and the Sprint Review are the only ceremonies where all the developers meet. Apart from that, all the other ceremonies are held with their respective Team (Team A, Team B).

The Backlog Refinement is held for one hour and has the purpose of refining tickets. The Team A has one Backlog Refinement per week, whilst Team B has two Backlog Refinements per week since they run in two-week iterations, their sprint backlog is very big, and they need more tickets to be ready for their Sprint.

The Ticketing session occurs once per week for one hour and is where the Teams look at the tickets refined in Backlog Refinement and estimate them.

After Sprint Review, the Team has Sprint Planning, beginning the cycle again.

# 6. CONCLUSIONS AND FUTURE RESEARCH

The main goal of this research was to show how useful Scrum can be to an organization when implemented well. This framework is rather useful when used correctly, but it is not the solution to every company, to every team or to every situation. Is important to understand the company environment and the culture surrounding it. In Living Map, Scrum helped the Teams delivering a very big project in a tight deadline. The Teams developed a deep understanding of the project, of the product that was being developed and achieved product increments every sprint.

Living Map has three distinct areas – Design, GIS and Development – and making these three areas connect with Scrum was extremely hard. Since Scrum was created to help software teams, this was the focus of the investigation. Although, at the same time, the researcher was also the Scrum Master of the GIS team, which had a higher maturity level and full understanding of the framework. The Design team abandoned Sprint and started using Kanban. The investigation did not focus on this area, but it was interesting seeing how sometimes, the problems are not about what type of framework you are using but rather, how well a Team works together. It does not matter if you use Scrum or Kanban, if the Team does not have discipline, no framework can help them.

Re-structuring Scrum and giving the Teams the capability of deciding on how and when to do a piece of work brought many benefits to the company. By using Scrum, the Teams were able to deliver a major project that involved many features and deadlines and intra-team dependencies, however, Scrum has a big hole on how to coordinate intra teams.

The points Planned, and the points Achieved had a steady increase, like the points per person per day. The Teams started focusing in other problems – such as code structure, training, best coding practices – instead of worrying about the Scrum process. The flow of the Sprint also improved, having tickets move into “Done” throughout the iteration rather than only on the last two days.

The transparency and visibility of the work that was being done was also improved by having joined Daily Scrum Meetings. The rule implemented with the “microphone” also had a positive outcome, reducing the length of the ceremony from 20-30 minutes for each team to 5-10 minutes for both teams. These meetings started being shorter and more efficient.

Using Scrum in Research and Development teams demands a lot of work because the tickets need a high level of refinement, the Teams must understand what is happening at all time and there are a lot of tickets that go from being relevant to irrelevant in a very short amount of time. On top of that, is hard to ask for detail or estimates on a piece of work that is unknown or is

still un-defined. Planning a Sprint that only lasts for one week also demands small tickets, otherwise there is not enough time for it to go through the ticket flow and for it to reach the "Definition of Done". The acceptance criteria on the tickets was not well defined, and many times the Team members in Team A had a lot of questions about the implementation of a ticket when in comparison to the Team B team members that were clear on what it was expected from every ticket in the Sprint.

Even though it requires a lot of work and organization, one-week Sprints helps the Teams to see the work and project in a whole way and it also helps to prioritize what is actually relevant, but adds pressure on the tester, and in making sure the tickets are ready in one week. What happened multiple times was that in the first instance of the sprint, most ticket would carry over to the second part – barely moving any tickets into "done" – and in the second part of the sprint a higher percentage of work would be released.

Letting the Teams refine their own tickets (by introducing Backlog Refinements) brought benefits in terms of project understanding, better integration of future and Team collaboration, however it also brought extra work. Team B had a Sprint Capacity of 90 story points by the end of Sprint VI. Keeping the Sprint Backlog with such a big amount of tickets was troublesome because it required more Backlog Refinements and Ticketing sessions.

In global, not only did the performance indicators grow significantly, but also, the understanding of Scrum increased, and the overall satisfaction of the Teams also improved. It should be said though, that Team B had a greater performance development when compared with Team A. Team B was focused in customer faced work that was easier to break down in tickets and had higher predictability being easier to plan and to estimate. Adding to this, Team B was working in the same office, whislt two senior members of Team A were working in London, in the other office, which creates communication barriers and difficultes ad-hoc conversations and brain storming. Team B was also thriving due to their Team Lead, since he would have frequent 1:1's with them, and had a vast understanding of Scrum and other Agile Methodologies.

Difficulties found when moving the Team A into one week Sprints varied from – not being able to complete Sprint – in the first part of the Sprint, the %Achieved was very low, having tickets roll over to the second part of the Sprint; not having enough time to refine, estimate and plan the next Sprint; managing tickets that went in to the Sprint but were no longer relevant after two days. Within a multi-disciplinary company, using Scrum with all the Teams has also raised some problems, since dependencies would make the project drag for longer than it was intended to.

Agile methodologies are such a vast and interesting subject, and more research should be done regarding it. The Scrum Guide draws guidelines that should be followed, but that are not mandatory. As it was seen in this investigation, some ceremonies were added to the Scrum flow to improve visibility and it should be noticed that something that works for a team might not work for another.

Future research should be done in Scrum and Research and Development Teams. How to structure work that does not have a release increment? How should a Team estimate and break down work that is still not detailed, and how should the Teams cope with the Sprint changes? Scrum protects the Teams perfectly, but it also creates silos of Teams, that will not help each other in case of dependencies, because they did not account for it in their Sprint Capacity. What is the best approach to account for this in-between work in intra Team collaboration? In Living Map, we added a timed-box Placeholder ticket to account for team collaboration, but is this a solution for the problem or is it rather a quick fix?

# REFERENCES


Accuware. (2019). About - Accuware. Retrieved January 26, 2019, from https://www.accuware.com/about/

Agarwal, M., & Majumdar, R. (2012). Tracking Scrum projects Tools, Metrics and Myths About Agile. *International Journal of Emerging Technology and Advanced Engineering*, *2*(3), 97–104.

Awad, M. A. (2005). *A Comparison between Agile and Traditional Software Development Methodologies*. Retrieved from http://pds10.egloos.com/pds/200808/13/85/A_comparision_between_Agile_and_Traditional_SW_development_methodologies.pdf

Bassil, Y. (2012). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering & Technology (IJET)*, *2*(5), 2049–3444. Retrieved from http://iet-journals.org/archive/2012/may_vol_2_no_5/255895133318216.pdf

Beck, K. (1999). *Embracing Change with Extreme Programming*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.617.9195&rep=rep1&type=pdf

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhacement. *Computer*, *21*(5), 61–72. Retrieved from http://www.dimap.ufrn.br/~jair/ES/artigos/SpiralModelBoehm.pdf

Chrisman, N., Wiley, J., Chichester, N. Y., Toronto, B., & Weinheim, S. (1999). *Exploring Geographic Information Systems*. Retrieved from https://pdfs.semanticscholar.org/1804/643869b1708eb604bdad2362b5e06b61fdab.pdf

Cohen, D., Lindvall, M., & Costa, P. (2003). An Introduction to Agile Methods. *Advances in Computers*, *62*. https://doi.org/10.1016/S0065-2458(03)62001-2

Cohn, M. (2005). *Agile Estimating and Planning*. Retrieved from http://synchronit.com/downloads/freebooks/Agile Estimating and Planning.pdf

ESRI. (2019). Who We Are | About Esri. Retrieved January 26, 2019, from https://www.esri.com/en-us/about/about-esri/who-we-are

Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. Retrieved from http://andrey.hristov.com/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf

Goodchild, M. F. (2006). *Approaches to Human Geography*. Sage Publications. Retrieved from http://www.repository.embuni.ac.ke/bitstream/handle/123456789/1277/%5BStuart_Aitken%2C_Gill_Valentine%5D_Approaches_to_Huma%28BookSee.org%29.pdf?sequence=1&isAllowed=y#page=262

Gould, J. D., & Lewis, C. (1985). Designing for Usability: Key Principles and What Designers Think. *Magazine Communications of the ACM*, *28*(3), 300–311. Retrieved from http://ldt.stanford.edu/~jsulzen/james-sulzen-portfolio/classes/ED229b/Syllabus/ED229B/www.stanford.edu/class/ed229b/fall00/readings/p300-gould.pdf

Highsmith, J. A. (2002). *Agile software development ecosystems*. Addison-Wesley. Retrieved from https://epdf.tips/agile-software-development-ecosystems.html

IEE Computer Society. (2014). *Guide Software Engineering Body of Knowledge*. (Pierre Bourque; & Richard E., Eds.). Retrieved from http://beamphys.triumf.ca/info/SWEBOKv3.pdf

Kelly, A. (2012). *An Agile Reader*. Software Strategy, Ltd.

LocusLabs. (2019). LocusLabs - Company. Retrieved January 26, 2019, from https://locuslabs.com/about/

Mapbox. (2019). About | Mapbox. Retrieved January 26, 2019, from https://www.mapbox.com/about/

Murugaiyan, D. (2012). Waterfall vs V-Model vs Agile: A comparative study on SDLC. *International Journal of Information and Business Management*, *2*(1), 26–30. Retrieved from www.jitbm.com

Norman, D. (2013). *The Design of everyday things*. Basic Books. Retrieved from www.basickbooks.com

Pointr. (2019). Company - Pointr. Retrieved January 26, 2019, from https://www.pointrlabs.com/pointr/

Raju, H. K., & Krishnegowda, Y. T. (2013). Kanban pull and flow - a transparent workflow for improved quality and productivity in software development. In *Fifth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013)* (pp. 44–51). Institution of Engineering and Technology. https://doi.org/10.1049/cp.2013.2233

Rasmusson, J. (2019). Burndown Charts. Retrieved February 1, 2019, from http://www.agilenutshell.com/burndown

Schwaber, K. (1996). The Origins of Scrum. *Controlled Chaos: Living on the Edge*. Retrieved from http://www.scrummanager.net/files/Living_on_the_Edge.pdf

Schwaber, K. (2001). *Agile Software Development with Scrum*. Prentice Hall PTR Upper Saddle River, NJ, USA ©2001. Retrieved from http://sutlib2.sut.ac.th/sut_contents/H129174.pdf

Schwaber, K. (2009). Agile Project Management with Scrum. In *Measuring and Improving Performance* (Vol. 7, p. 163). CRC Press. https://doi.org/10.1201/9781420084191-c2

Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide™ - The Definitive Guide to Scrum: The Rules of the Game*. Retrieved from https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

Serie, C. (2010). *You and Your Action Research Project* (Vol. 2). https://doi.org/10.4324/9780203281291

Sommerville, I., Columbus, B., New, I., San, Y., Upper, F., River, S., … Tokyo, T. (2011). *Software Engineering* (Ninth). Sommerville. Retrieved from http://www.uoitc.edu.iq/images/documents/informatics-institute/exam_materials/Software Engineering (9th Edition) by Ian Sommerville.pdf

Sugimori, Y., Kusunoki, K., Cho, F., & Uchikawa, S. (1977). Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system. *The International Journal of Production Research*, *15*(6), 553–564. https://doi.org/10.1080/00207547708943149

Sutton, S. M. (2018). Informed projection: Using What You Know to Make Simple Estimates of Work Better. In *Proceedings of the 2018 International Conference on Software and System Process - ICSSP '18* (pp. 76–85). New York, New York, USA: ACM Press. https://doi.org/10.1145/3202710.3203147

Technavio. (2017). *Global Digital Map Market 2017-2021*. Retrieved from https://www.marketresearch.com/Infiniti-Research-Limited-v2680/Global-Digital-Map-11310909/

Tereso, A., Ribeiro, P., Fernandes, G., Loureiro, I., & Ferreira, M. (2018). Project Management Practices in Private Organizations. *Project Management Journal*, *50*(1), 6–22. https://doi.org/10.1177/8756972818810966

Usman, M., Mendes, E., Weidt, F., & Britto, R. (2014). Effort Estimation in Agile Software Development: A Systematic Literature Review. *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, 82–91. https://doi.org/10.1145/2639490.2639503

# APPENDIX I - DATA COLLECTED PER SPRINT

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint A | 45 | 35.5 | 78.9% | 9 | 3.94 | 44.8 | 0.79 |
| B | Sprint A | 62.5 | 52.5 | 84.0% | 9 | 5.83 | 26.6 | 1.97 |
| Joined | Sprint A | 107.5 | 88 | 81.9% | 9 | 9.78 | 71.4 | 1.23 |

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint B | 46 | 31 | 67.4% | 8.5 | 3.65 | 23.45 | 1.32 |
| B | Sprint B | 46 | 20.5 | 44.6% | 8.5 | 2.41 | 23.37 | 0.88 |
| Joined | Sprint B | 92 | 51.5 | 56.0% | 8.5 | 6.06 | 46.82 | 1.10 |

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint C | 38.5 | 36 | 93.5% | 9.5 | 3.79 | 20.1 | 1.79 |
| B | Sprint C | 62 | 31 | 50.0% | 9.5 | 3.26 | 21.85 | 1.42 |
| Joined | Sprint C | 100.5 | 67 | 66.7% | 9.5 | 7.05 | 41.95 | 1.60 |

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint I | 41 | 16 | 39.0% | 5 | 3.20 | 23.6 | 0.68 |
| B | Sprint I | 63 | 62 | 98.4% | 9 | 6.89 | 20.45 | 3.03 |
| Joined | Sprint I | 106 | 114.5 | 108.0% | 9 | 12.72 | 49.35 | 2.32 |

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint II | 68 | 52 | 76.5% | 9 | 5.78 | 40 | 1.30 |
| B | Sprint II | 54 | 57.5 | 106.5% | 9 | 6.39 | 24.25 | 2.37 |
| Joined | Sprint II | 122 | 109.5 | 89.8% | 9 | 12.17 | 64.25 | 1.70 |

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint III | 43 | 46.5 | 108.1% | 5 | 9.30 | 24 | 1.94 |
| A | Sprint III | 40 | 19 | 47.5% | 4 | 4.75 | 20 | 0.95 |
| B | Sprint III | 66 | 59 | 89.4% | 9 | 6.56 | 24.7 | 2.39 |
| Joined | Sprint III | 149 | 124.5 | 83.6% | 9 | 13.83 | 68.7 | 1.81 |

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint IV | 39.5 | 20 | 50.6% | 5 | 4.00 | 24.2 | 0.83 |
| A | Sprint IV | 33 | 28 | 84.8% | 4 | 7.00 | 21.49 | 1.30 |
| B | Sprint IV | 78.5 | 61 | 77.7% | 9 | 6.78 | 33 | 1.85 |
| Joined | Sprint IV | 151 | 109 | 72.2% | 9 | 12.11 | 77.2 | 1.41 |

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint V | 41 | 16 | 39.0% | 5 | 3.20 | 23.6 | 0.68 |
| A | Sprint V | 46 | 40 | 87.0% | 4 | 10.00 | 18.9 | 2.12 |
| B | Sprint V | 88.5 | 77.5 | 87.6% | 9 | 8.61 | 33.3 | 2.33 |
| Joined | Sprint V | 175.5 | 133.5 | 76.1% | 9 | 75.80 | 78.99 | 1.69 |

| Team | Sprint | Planned | Achieved | % Achieved | Days in Sprint | Points/Day | PersonDays | Points/person/day |
|---|---|---|---|---|---|---|---|---|
| A | Sprint VI | 34 | 23.5 | 69.1% | 5 | 4.70 | 20 | 1.18 |
| A | Sprint VI | 40 | 40 | 100.0% | 4 | 10.00 | 17 | 2.35 |
| B | Sprint VI | 91 | 91 | 100.0% | 9 | 10.11 | 38.7 | 2.35 |
| Joined | Sprint VI | 165 | 154.5 | 93.6% | 9 | 53.40 | 75.7 | 2.04 |

# APPENDIX II - COMPLEMENTARY SPREADSHEET

| | |
|---|---|
| HoursPerDay | 5 |
| DaysInSprint | 5 |
| Hours:Points | 0.28 |

| | Days Absent | % Availability | PersonDays | Points/Person |
|---|---|---|---|---|
| Member 1 | 0.5 | 50 | 2.25 | 3.15 |
| Member 2 | 0 | 70 | 3.5 | 4.9 |
| Member 3 | 2 | 70 | 2.1 | 2.94 |
| Member 4 | 0 | 50 | 2.5 | 3.5 |
| Member 5 | 0 | 55 | 2.75 | 3.85 |
| Member 6 | 0 | 65 | 3.25 | 4.55 |
| Member 7 | 1 | 65 | 2.6 | 3.64 |

| Total person days | Story Points | Points/Person/day | Team Commitment |
|---|---|---|---|
| 18.95 | 26.53 | 1.4 | 26 |
| | | | Points In Jira |
| | | | 40.5 |

| Carry Over | | | |
|---|---|---|---|
| Ticket | Original | Remaining | Carry Over |
| pps-308 | | 0 | 0 |
| vm-241 | 3 | 0 | 3 |
| DCA-33 | 3 | 1 | 2 |
| dca-34 | 1 | 0.5 | 0.5 |
| PPS-317 | 5 | 1 | 4 |
| PPS-320 | 2 | 0.5 | 1.5 |
| **PPS-234** | **3** | **0.5** | 2.5 |
| pps-305 | 3 | 2 | 1 |
| | | | |
| | | | 14.5 |

| | |
|---|---|
| HoursPerDay | 5 |
| DaysInSprint | 9 |
| Hours:Points | 0.47 |

| | Days Absent | % Availability | PersonDays | Points/Person |
|---|---|---|---|---|
| Member 1 | 2 | 60 | 4.2 | 9.87 |
| Member 2 | 3 | 80 | 4.8 | 11.28 |
| Member 3 | 0 | 90 | 8.1 | 19.035 |
| Member 4 | 0 | 70 | 6.3 | 14.805 |
| Member 5 | 0 | 70 | 6.3 | 14.805 |
| Member 6 | 0 | 40 | 3.6 | 8.46 |

| Total person days | Story Points | Points/Person/day | Team Commitment |
|---|---|---|---|
| 33.3 | 78.255 | 2.4 | 80 |
| | | | Points In Jira |
| | | | 89 |

| Carry Over | | | |
|---|---|---|---|
| Ticket | Original | Remaining | Carry Over |
| VM-250 | 5 | 0 | 5 |
| MAS-256 | 3 | 3 | 0 |
| MAS-257 | 3 | 3 | 0 |
| VM-197 | 2 | 0 | 2 |
| SA-112 | 3 | 1 | 2 |
| | | | 0 |
| | | | **9** |

# ANNEX I

## Scrum Quizz – taken from Scrum Master Certificate Course

When does a Development Team member become the sole owner of a Sprint Backlog item?

1. At the Sprint planning meeting.
2. Never. All Sprint Backlog Items are "owned" by the entire Development Team, even though each one may be implemented by an individual development team member.
3. Whenever a team member can accommodate more work.
4. During the Daily Scrum.

The CEO asks the Development Team to add a "very important" item to a Sprint that is in progress. What should the Development Team do?

1. Add the item to the current Sprint without any adjustments.
2. Add the item to the current Sprint and drop an item of equal size.
3. Add the item to the next Sprint.
4. Inform the Product Owner so he/she can work with the CEO.

Who is on the Scrum Team?

1. The Scrum Master.
2. The Product Owner.
3. The Development Team.
4. Project Manager.
5. None of the above.

Why is the Daily Scrum held at the same time and same place?

1. The place can be named.
2. The consistency reduces complexity.
3. The Product Owner demands it.
4. Rooms are hard to book, and this lets it be booked in advance.

The Development Team should not be interrupted during the Sprint. The Sprint Goal should remain intact. These are conditions that foster creativity, quality and productivity. Based on this, which of the following is **FALSE**

1. The Product Owner can help clarify or optimize the Sprint when asked by the Development Team.
2. The Sprint Backlog is fully formulated in the Sprint Planning meeting and does not change during the Sprint.
3. As a decomposition of the selected Product Backlog Items, the Sprint Backlog changes and may grow as the work emerges.
4. The Development Team may work with the Product Owner to remove or add work if it finds it has more or less capacity than it expected.

Which statement best describes a Product Owner's responsibility?

1. Optimizing the value of the work the Development Team does.
2. Directing the Development Team.
3. Managing the project and ensuring that the work meets the commitments to the stakeholders.
4. Keeping stakeholders at bay.

During the Daily Scrum, the Scrum Master's role is to:

1. Lead the discussions of the Development Team.
2. Make sure that all 3 questions have been answered.
3. Manage the meeting in a way that each team member has a chance to speak.
4. Teach the Development Team to keep the Daily Scrum within the 15 minute time-box.
5. All answers apply.

Which statement best describes the Sprint Review?

1. It is a mechanism to control the Development Team's activities during a Sprint.
2. It is when the Scrum Team and stakeholders inspect the outcome of a Sprint and figure out what to do next.
3. It is a demo at the end of the Sprint for everyone in the organization to check on the work done.

The three pillars of empirical process control are:

1. Respect For People, Kaizen, Eliminating Waste.
2. Planning, Demonstration, Retrospective.
3. Inspection, Transparency, Adaptation.

4. Planning, Inspection, Adaptation.
5. Transparency, Eliminating Waste, Kaizen.

How much work must a Development Team do to a Product Backlog item it selects for a Sprint?

1. As much as it has told the Product Owner will be done for every Product Backlog item it selects in conformance with the definition of "Done".
2. As much as it can fit into the Sprint.
3. All development work and at least some testing.
4. Analysis, design, programming, testing and documentation.

Which one of the following is not a Scrum artifact?

1. Product Backlog.
2. Story.
3. Sprint Backlog.
4. Burndown Chart.

“Velocity” is a measure of …

1. Customer buy-in.
2. Project progress.
3. Executive support.
4. Risk mitigation.

Scrum Teams are (select the best option):

1. Self-organizing, Cross-functional.
2. Cross-productive, Self-organizing.
3. Self-destructive, Self-motivating.

Why do we use the Fibonacci Sequence to estimate Stories?

What are the three questions a user story should answer?

What is the name of the method we use to estimate tickets?