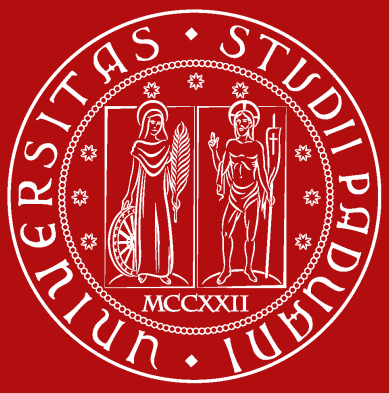

DIPARTIMENTO
MATEMATICA

Supporting tool for Agile software development

Experience from a real use case

Bachelor's Degree thesis in Computer Science

Candidate: Ciprian Stefan Voinea
Student ID: 1143057

Supervisor: Dott. Armir Bujari

Accademic Year 2018 - 2019

Università degli Studi di Padova

Department of Mathematics "Tullio Levi Civita"

Bachelor Thesis in Computer Science

Supporting tools for Agile software development: experience from a real use case

Supervisor
Dott. Armir Bujari

Candidate
Ciprian Stefan Voinea
Student ID
1143057

Accademic Year 2018 - 2019

To my parents,
for the sacrifices they made for me

Abstract

ABSTRACT

Sommario

SOMMARIO

Contents

Listing of figures

1 Introduction

INTRODUCTION TO WHAT IS A WAY OF WORKING
HOW IT IS APPLIED IN MODERN SOFTWARE DEVELOPMENT COMPANIES
THE IMPORTANCE OF HAVING A GOOD WAY OF WORKING FOR A COMPANY IN ORDER TO BE PRODUCTIVE

1.1 Premise

This document is a report of the two month curricular internship done between June and July 2019 at Athonet under the supervision of Dr. Fabio Giust. It contains a description of the work that I have done and an introduction to Agile and Scrum software developing methodologies.
To introduce or to describe some of the arguments I will be using some comic strips of Dilbert, a character invented by Scott Adams[1]. It satirically represents the problems that can be present in any small or big IT company.

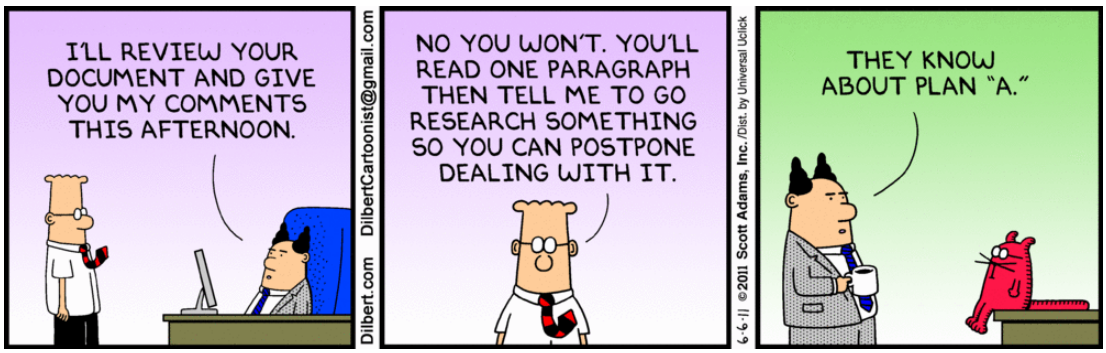

Figure 1.1: Dilbert, "*Plan A*", Monday June 06, 2011

The document contains a Glossary section that describes technical words that would need a more specific introduction: these are marked as EXAMPLE$_G$.

1.2 THE COMPANY

Athonet[2] is a telecommunication company headquartered nearby Vicenza, Italy. It stem from the idea that broadband networking should be easily available to people in rural areas and to companies that work in mission critical services or special environments like shipping, mining companies or even hospitals (safety critical).
Officially it was started in 2004, although a working prototype of their idea was already developed by the CEO (Chief Executive Officer) and CTO (Chief Technology Officer) that were working alongside in Ericsson[3] at that time.

Figure 1.2: Athonet's logo

Low latency communication, reliability and security are at the core of what Athonet provides. Their main product is PriMo, a device that allows to create a dedicated core network, or enterprise LTE.

Figure 1.3: Athonet's main product **"*PriMo*"**[4]

The philosophy behind PriMo is that "*the software is the network*", as Athonet was a pioneer in decoupling the software from the hardware, in order to have the solution running on different general purpose computing hardware, e.g., based on x86 processors.
In 2012, after a disastrous earthquake destroyed all the communication lines in Emilia Romagna, Athonet has reestablished connectivity, thus showing the how PriMo can be used on the field in emergency situations.
Athonet installed PriMo at the top of a school, allowing them to cover the affected area not only for the operators of Servizio Civile but for the citizens as well.
In December 2013, Athonet has been rewarded by Giorgio Napolitano, the then President of the Italian Republic, with a medal for merits for the sustainability in the digital sector[5].
Lately they have migrated some of their functions to the cloud: by using AWS they have achieved a hybrid product, BubbleCloud, a plug and play solution that allows to locally deploy the physical Edge Nodes while managing them from the AWS cloud.
This small business has much to offer, considering that some of it's competitors are giants like Nokia and Ericsson.
As a proof of the continuous will to improve and for the solution it provides, Athonet has been awarded with four Global Mobile Awards at the GSMA Mobile World Congress, held in February 2019 in Barcelona[6].

1.3 The project

It consists in installing and configuring two main software tools, Jira and Confluence, alongside plugins to add more functions and enhance their potentiality. However, the most important part of this project is not installing the software, but adapting it to the needs of the company.
As said earlier Athonet my still be small, but it's a growing company, and because of this it needs to give itself some internal rules and specifications to follow when working on a task, communicating with the client or even share internal information to employees. Not only for themselves but for clients they work with as well, since most big companies require that their partners have internal regulations, no matter the number of people in the company.
As I will explain in the following chapters Jira is an Issue Tracking System, a software that allows to follow the tasks (like resolving bugs or implementing features) that are related to a project, while Confluence is a software for sharing knowledge, that means sharing internal documents, keeping a wiki, having documentation available and even for customers.
My task was to demonstrate that these tools are what Athonet needs to be a strong an coherent company, where information is always shared and available, while maintaining a history of changes, by creating an environment that suited their needs and that can evolve alongside the company.
To do this I have learned the basics of Agile and Scrum methodologies, how a company operates internally and with their clients and how introducing a new tool may improve the way of working, even if it creates chaos at the beginning.

1.4 Document organization

This thesis is organized as follows:

- Chapter 1 or "*Introduction*": describes the overall content of this document
- Chapter 2 or "*The internship project*": describes in detail the objectives and planning of the internship project
- Chapter 3 Chapter 3 or "*Agile processes and methodologies*": an introduction to the Agile software development
- Chapter 4 or "*Jira and Confluence: the essentials*": describes the most valuable functionalities of Jira and Confluence

- Chapter 5 or "*Project implementation*": details how the project has been implemented by dividing it into time periods
- Chapter 6 or "*Conclusions*": contains the retrospective of the project, future developments and personal considerations

A goal without a plan is just a wish.

2 The project

This chapter contains the results of the discussions and planning that have been made prior to the beginning of the internship. These contents are described in more detail in the "*Piano di Lavoro*" (work plan in English), a document that contains an estimation of resources for each task that composes the project.
At the beginning of May I met with the tutor to understand the needs of the company and draft a plan. Below I describe how a resolution for the problem was first thought.

2.1 The company's needs

At the time of writing it has nearly 40 employees and it is expected to hire new people soon. Athonet uses multiple software tools for keeping track of projects, sharing information internally and with the clients, visualizing product roadmaps, etc.
The most important tools they are currently relying on are:

- Redmine[7] as an Issue Tracking System
- SharePoint[8] as a collaborative platform
- Planner[9] for visualizing project roadmaps
- GitLab[10] as a repository

While some of these packages are compatible, others were not developed to be used together, although there are plugins that allow to interconnect them. These don't allow much customization since they are programmed by third party developers and not always the compatibility is up to date with the latest releases.
There is a need then to provide employees with a suite of stable tools that are easily interconnected and with a vast and well done documentation. Also, these tools must be easy to maintain and update, since they don't have to become obsolete and outdated.
Nobody likes legacy systems.

Figure 2.1: Dilbert, "*Plan A*", Monday June 06, 2011

Internal changes may not be directly visible to the clients, but the effect of having a much organized company, where there is always track of the work done and in progress ensures that when there is a request from the client it does not go unseen or unanswered.
Forcing employees to use a software instead of another one though is not easy without creating chaos. This is why a good and easy to consult documentation is the key on helping employees transition.

2.2 Requirements and objectives

The final objective was to create a suitable and working environment that would be ready to transfer it into production and explain the users how to use it.
Requirements can be divided in three categories based on their relevance:

- Mandatory ("*Obbligatorio*" in italian): that have to be implemented, represent the core of the project;
- Desirable ("*Desiderabile*" in italian): not mandatory for the final objective but add greater value;

- Optional ("*Facoltativo*" in italian): add value to the project but not as much as the previous ones, carried out only if there is time left for them.

Each requirement has a unique ID that is composed by the first letter of their importance (from the italian word to resemble the "*Piano di Lavoro*") and an increasing number.
A detailed list of requirements that describe more accurately the ones included in the original planning document can be found in B.

2.3 Time division and planning

To formalize the time division of the project I have used a Gantt diagram and a table contains a realistic approximation of the hours spent per task.
The internship was planned for a duration of ten weeks and the project has been spread such that I could have time to understand the tools and use them to get acquainted. This time period also includes the phases for getting feedback from users, fine tune the configuration, explain the tools to various members of the company and, in case of incidents, slack time.
The complete Gantt diagram can be found in A.1.
I started to create a temporal sequence only after I well understood the requirements and got an idea about how Atlassian's software works, also I wanted to figure out the final objective and thinking how to achieve it. As it can be seen in the Gantt diagram, there are some tasks that overlap in time: for example the Study of the Atlassian suite overlaps with Configuration of the environment tools. This because the tasks may have something in common or one helps the achievement of the other. While studying the requirements for the Atlassian software I decided I would also configure the host machine.
I have coarse grained planned four main time periods:

- Learning
- Implementing
- Testing and fine tuning
- Feedback

Each of these contain smaller tasks that involve studying the tools, installing and configuring them, etc.
As discussed with the tutor, meetings had to be considered so that I could explain the progress

to him and other company figures that would eventually use the software.
As milestones I considered having a deliverable that could be easily transitioned from the development to the production environment, such as a script that installs and configures the software or a container. By definition this must be versionable and with a well written changelog.

2.4 Approaching the problem

To understand better the potentiality of the tools, I have thought of some main use cases that, according to the requirements in ..., could indicate the main figures that interact with the tools and how. Write that I have thought of use cases and associate them to the requirement?

If a team couldn't be fed with two pizzas, it was too big.

Jeff Bezos

3

Agile Software Development

Before getting into the implementation and adaptation of Jira and Confluence, let's back up a little and understand the concept of Agile, a Software Development Life Cycle (or SDLC) model.

SDLCs did not emerge until the 1960s; they are considered to be the oldest formalization of framework. A SDLC refers to the ensemble of activities that compose a software project.

It starts with the concepts of understanding the problem as well as the requirements and it ends with the retirement of the system (when there is no more maintenance) or with the cancellation of the project.

Small projects (generally for a single person) have a simpler life cycle: find the problem and write a program to solve it: once the problem has been solved, the program can be deleted and forgotten.

In larger projects, that require a team to be developed, there must be some explicit rules to set a higher quality for the software.

As activities are assigned to different people, it becomes critical that all participants share a common view of the execution of the project. A SDLC model is a framework providing the ordering and dependencies of life cycle activities, managing these can be a major impact for a successful project and its duration.

For example, a change in requirements during implementation may invalidate a substantial amount of work and delay the delivery of the system by several months. Different life cycle models prescribe different actions to handle such changes[11].

There are many SDLCs like Waterfall, Prototyping, Iterative and Incremental Development, Spiral Development, Rapid Application Development, and Extreme Programming. (XP).

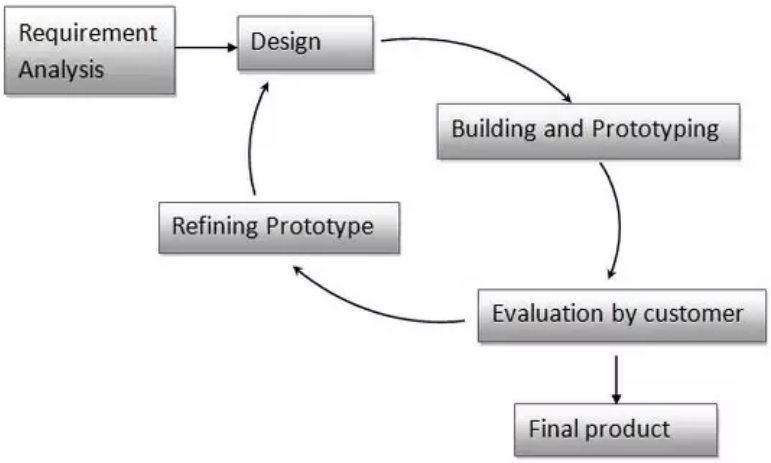

Figure 3.1: The Prototype model

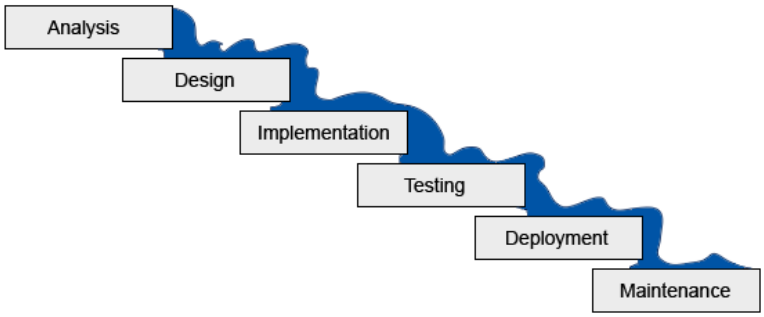

Figure 3.2: The Waterfall model

As the word "*model*" suggests, each company may have its own SDLC designed ad hoc for their internal use.
This led to the creation of a standard that presents the guidelines for the elements that are most frequently used in describing a process: the title, purpose, outcomes, activities, task and information item. The ISO/IEC TR 24774:2010: Systems and software engineering – Life cycle management – Guidelines for process description[12].
The complexity and slowness in producing a concrete product in older SDLCs brought the need for a faster and more communicative model.

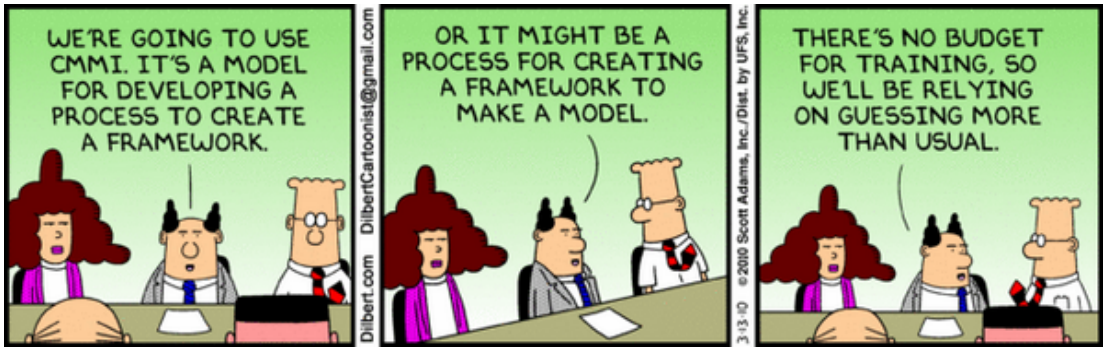

Figure 3.3: *Redmine*'s logo

This chapter describes the most fundamental points of the Agile method, how it started and the adaptations that derived from it, like Scrum. At the end, it also explains how Athonet's adaptation of the Agile life cycle works.

3.1 The need for a new Software Life Cycle

The decade of 1990 represented a very important turning point for the digital industry. Computers were spreading everywhere and the software companies faced the so-called application development crisis.
The problem was that businesses moved too fast and within the space of three years, requirements, systems, and even entire businesses were likely to change. It meant that a lot of software ended up being incomplete or canceled halfway and the ones that made it through, even if they fulfilled the original objectives of the client, may not meet all the business needs[13].
SDLC models can be of two types:

- Iterative: enhances the evolving versions until the complete system is implemented and ready to be deployed
- Incremental: the product is designed, implemented and tested incrementally until it is finished

One of the most characteristic models used before Agile was the Sequential model (or Waterfall). The waterfall model became very famous because it has many strong points as:

- Uses clear structure
- Determines the end goal early

- Transfers information well

On the other hand, it became obsolete when projects started to be more dynamic and complex. Some of it's disadvantages are:

- Makes changes difficult
- Excludes the client and/or end user
- Delays testing until after completion

The excessive documentation, the forceful binding to the unchangeable decisions made early in the project and the little communication with the client brought to the need of a new model that prioritizes the product and the stakeholders over bureaucracy.
Those things frustrated people like Jon Kern, an aerospace engineer in the 1990s that with other figures from different industries "*were looking for something that was more timely and responsive*", as he noted. He was one of 17 software leaders that started meeting informally and talking about ways to develop software in a simpler way without the excess of documentation and other strict rules.
These talks led to the now famous Snowbird meeting (in Utah, February 2001), when the Agile Manifesto was written down and published.

3.2 The Agile manifesto

The Agile Manifesto is a brief document built on four foundational values and twelve supporting principles for Agile software development[14].
The four values written on the official website[15] are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The responsiveness of people and embracing the importance of changes are the fundamentals of Agile. Although documentation is secondary, it's important to note that Agile streamlines documentation and does not eliminate it.
These twelve principles emphasize things like "*early and continuous delivery of valuable software" and "continuous attention to technical excellence*", and are:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

All of them are important but, in my opinion, the ones that add the most value to the Agile line of thought and that differentiate it from the other methods are 2, 4 and 6: they represent the intent of placing the product and the customer above everything else, allowing the use of small informal meetings (even if the decisions should be recorded) and the easy change of requirements because the client is always involved, even as an end user (tester).
Each Agile methodology applies the four values in different ways. However, all of them rely on these values to guide the development and delivery of high-quality, working software[16].

3.3 Agile's little big cousins

While Agile's manifesto contains values and principles, these are not prescriptive. In fact the manifesto does not outline specific processes, procedures or best practices. The goal is not to develop a rigid framework but rather create a mindset for software development. Agile is a blanket term that describes a set of software development principles.
There are many methodologies that derive from Agile's thinking, the most famous ones, according to the annual survey[17] from VersionOne's team are:

- Scrum
- Scrumban
- Kanban
- Extreme Programming (XP)

The essence of Scrum is being agile (fast): a small team that is highly flexible and adaptive.

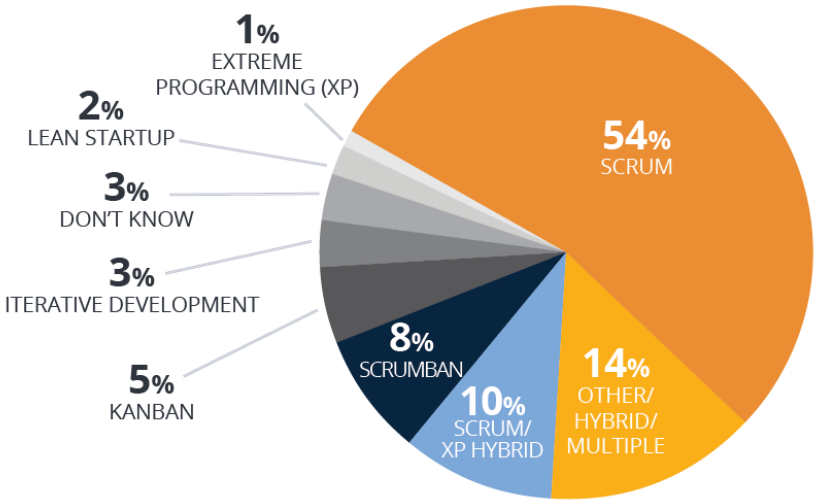

Figure 3.4: The Waterfall and Prototype SDLC models

Figure 3.5: The Waterfall and Prototype SDLC models

This survey is quite interesting because it provides information from small and big real companies that want to share their experience. A very important fact to be noted is that although Technology companies are the ones that mostly participated to the survey, there are other industries that use Agile and are interested in sharing their experience.

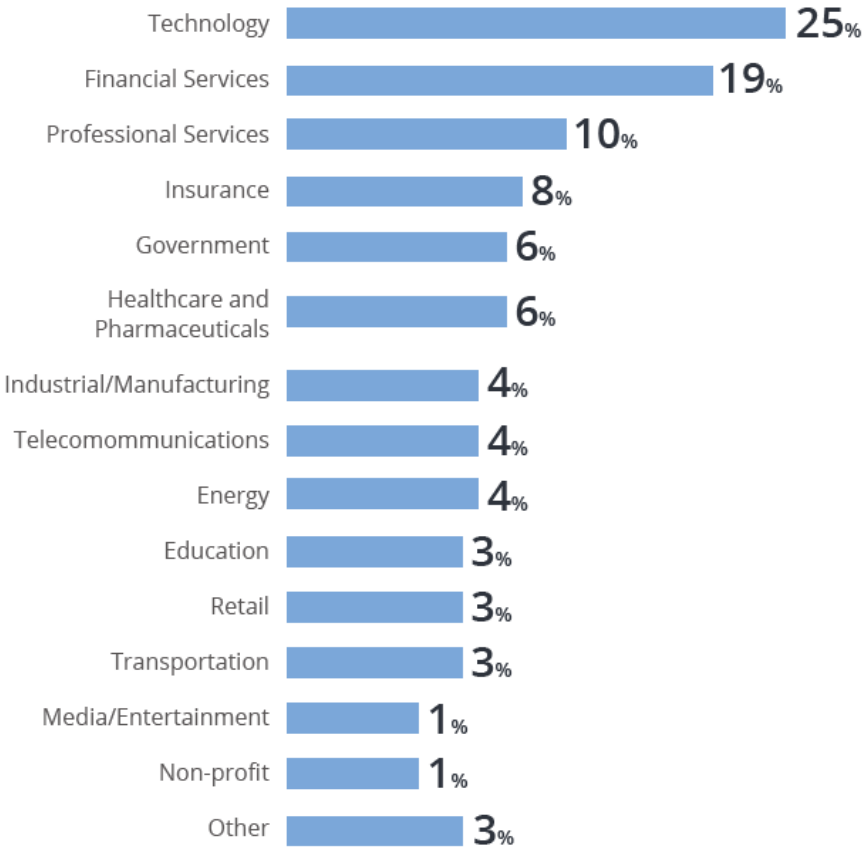

Figure 3.6: The Waterfall and Prototype SDLC models

It states that for the year 2018 (which marked their 13th annual report) the reasons for adopting Agile were productivity, improving team morale, reducing product risk (with a lesser percentage than the previous year) and about reducing project costs.
The measures of success mostly cited by the respondents were customer, or user, satisfaction and business value.

According to these companies, the benefits of adopting Agile are:

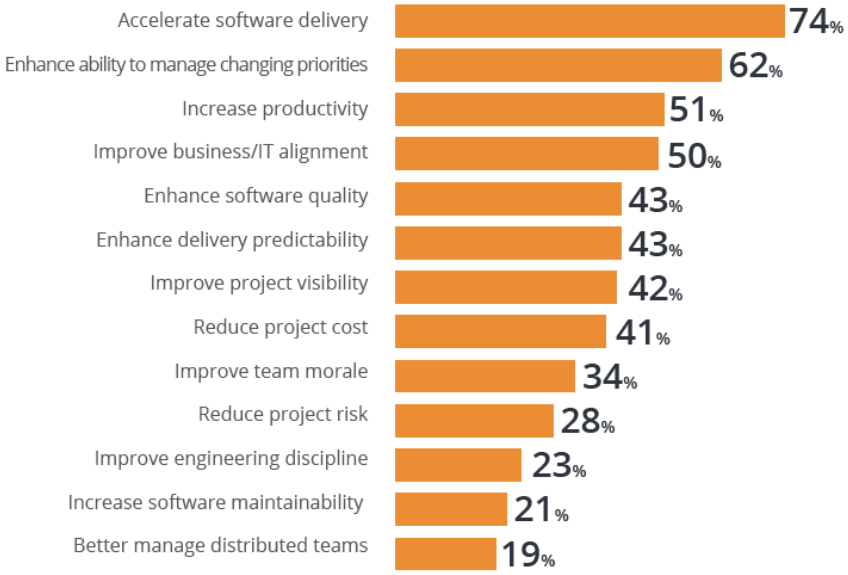

Figure 3.7: The Waterfall and Prototype SDLC models

Also the questionnaire contained a question about what are the recommended Agile project management tools.

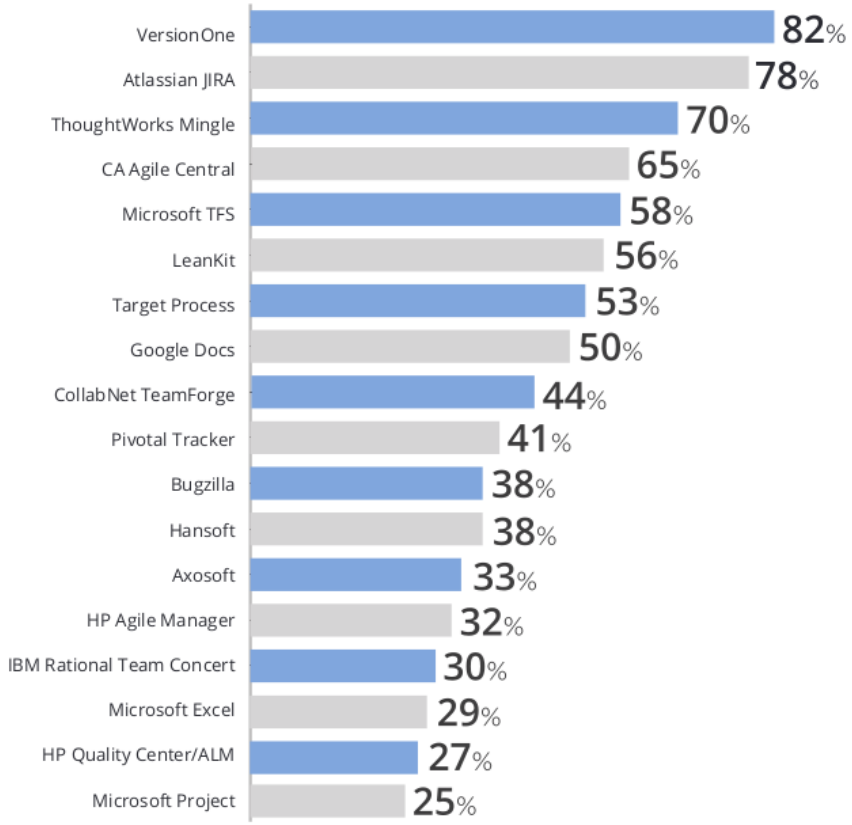

Figure 3.8: The Waterfall and Prototype SDLC models

As we can see Jira is the second most recommended one.

Figure 3.9: The Waterfall and Prototype SDLC models

3.4 Agile in practice

Briefly describe how some companies use Agile: Spotify / Netflix / Amazon

3.5 The roles in Agile

A distinctive characteristic of the Agile methodology is it's definition of roles: they are not positions, any given person takes on one or more roles and can switch them over time, and any given role may have zero or more people in it at any given point in a project[18]. The ideal team is considered to be composed of five or six people.
These roles are:

- Team leader: team coach or project lead in other methods (Scrum-master e.g.), he is responsible for facilitating the team, obtaining resources for it, and protecting it from problems
- Product owner: an executive or key stakeholder, the Product Owner has a vision for the end product and a sense of how it will fit into the company's long-term goal
- Team member: developer or programmer, is responsible for the creation and delivery of a system
- Stakeholder: any other person that has direct or indirect interest in the project

3.6 Time cycles and metrics

Briefly describe the most important time periods in Agile

3.7 Disadvantages of Agile Software Development

Despite the benefits offered by the Agile model, transition a company's way of working to it is not that easy and if done wrong it may make damage instead of good. According to the American entrepreneur Adam Fridman, here are the drawbacks[19] of Agile:

1. Less predictability: developers may not be able to quantify the full extend of the required effort
2. More time and commitment: a constant interaction, with many face-to-face conversations, is required
3. Greater demands on developers and clients: extensive user involvement that impacts the quality and success of the project

4. Lack of necessary documentation: new members that join the team may need more time to understand the project

5. Project easily falls off track: if a consumer's feedback or communications are not clear, a developer might focus on the wrong areas of development

3.8 What Agile variant Athonet uses

As many small companies, Athonet will not strictly use one Agile implementation. This is also due because of the nature of their product that is not released in simple software patches but it presents itself in a more monolithic version. After a research for the available software development life-cycle tools on the market, they chose to try Atlassian's Jira in tandem with Confluence.
As I will say in Chapter 5, the managers liked the idea of Kanban because it allows the employees to come in the morning and choose what they want to work on from the backlog without giving them a two week period of time but letting them complete the tasks in time for the following release. But they also liked the idea of having a tool that could transition from a type of project to another in case they want to start applying stricter rules in the future.

Figure 3.10: The Waterfall and Prototype SDLC models

Nulla facilisi. In vel sem. Morbi id urna in diam dignissim feugiat. Proin molestie tortor eu velit. Aliquam erat volutpat. Nullam ultrices, diam tempus vulputate egestas, eros pede varius leo.

Quoteauthor Lastname

4

Jira and Confluence: the essentials

Now that we have understood the concept of Agile, let's get to know Jira and Confluence, the software chosen by Athonet to implement it. ADT$_G$ These are proprietary tools developed and maintained by the Australian company Atlassian.

Figure 4.1: The logos of *Atlassian*, *Jira*, *Confluence* and *Jira Service Desk*

Jira is an Issue Tracking System that was first released in 2002, it's name is a truncation of Gojira, the Japanese word for Godzilla. This is a reference to another ITSs that was dominating the market at the time, Bugzilla. Now the competitors of Jira are other software like, for example, Redmine, VersionOne, PivotalTracker, Workzone or integrated ITSs in repositories like GitHub's or GitLab's issue trackers.
Athonet's previous choice was Redmine because it's open source (this implies it's free of commision), it has a medium-large community of people that use it and maintain it behind and

the plugins allow the integration with other tools used internally like repositories or, for example, software for reporting customer requests.

Figure 4.2: *Redmine*'s logo

Confluence, on the other hand, is designed as a collaboration platform for sharing knowledge like documents, product specifications, meeting notes and can be used as a wiki for internal use or for releasing information to the clients. Atlassian released the first version of Confluence in 2004, saying its purpose was to build "*an application that was built to the requirements of an enterprise knowledge management system, without losing the essential, powerful simplicity of the wiki in the process*"[20].
Since the first releases of these products, Atlassian has developed and acquired new tools like Bamboo, Clover, Crowd, Crucible, and FishEye, all orientated towards collaboration, content sharing, issue tracking, time scheduler, etc.
Both Jira and Confluence are written in Java.

4.1 Understanding what they can do

These tools are made to be integrated with one another, not only because they are made by the same company but they are strictly correlated.
Integrating an issue tracker with a platform able to share documents and thoughts allows a more granular analysis of the project requirements. This means that the company can store meeting notes and documents related to the project in Confluence, and when they are ready to move into the development phase, convert them to Issues in Jira.
Plugins can extend by far the usage and integration with other tools and the Atlassian Marketplace is full of them. Although there is a license that has to be paid, if used correctly these tools can allow the company to save money that could be lost in badly managed resources like time, documentation and even people.
Let's understand them better.

4.1.1 Jira

Over the past years Jira has become such an important software that Atlassian had to separate it in three specialized components, each with it's own scope. Here is a table from Jira's official documentation that contains all the major differences:

	JIRA Core	JIRA Software	JIRA Service Desk
Simple business projects	✓	✓	✓
Workflow editing	✓	✓	✓
Powerful issue search	✓	✓	✓
Dashboards	✓	✓	✓
Basic reporting	✓	✓	✓
Agile boards		✓	
Backlog planning		✓	
Agile reports		✓	
Development tool integration		✓	
Release hub		✓	
Queues			✓
SLAs			✓
Confluence knowledge base integration			✓
Detailed service metrics			✓

Figure 4.3: *Redmine*'s logo

I'll be describing some of the key features and purposes of each software, but a more in depth comparison can be found at:

Jira Core

Jira Core's main purpose is to handle business processes.

Jira Software

Jira Software's main purpose is to ho handle software projects.

Jira Service desk

Jira Service desk's main purpose is to handle customer requests

Jira Portfolio Plugin

Jira Portfolio was first designed as a plugin from ... then it was acquired by atlassian becaues
It's main purpose is to visualize project roadmaps

4.1.2 Confluence

As described earlier, Confluence is a collaboration platform. It allows to create spaces of different categories that can be associated with Jira projects.
SAY THAT THERE ARE MANY DIFFERENT SPACES + TELL DIFFERENCE

4.2 Key concepts for Jira

Jira has it's own terminology that needs to be understood in order to completely master the software. The key terms that must be known, according to ... are:

- Issues:
- Projects:
- Workflows:

4.3 How Athonet uses them

As I anticipated, Athonet has chosen Atlassian's Jira and Confluence because they need a single solution that is coherent and that could provide them a unique access for all the company figures to the projects and their related documentation.

impossibile sostituire certi tool come word per la creazione di documenti per la condivizione con clienti / utenti

These tools are complex, it is necessary to understand the scenario they can be used in and how to transition from the old software.
It is important for the company to adapt the tools for it and not change their entire way of working to accommodate to the new software.

4.3.1 Development

Compared to Redmine Jira has many more useful functionalities and on top of that it has a richer UI (User Interface). The development team will be the one that will use it the most, and because of that it is important that it can integrate with GitLab. With the GitLab Listener plugin Developers can interact with Jira's issues from the messages in the repository's operation.

4.3.2 Management

The most important feature the management department needed was the live roadmap. It's a key element for this kind of company since it allows to keep an eye the trend in development and manage the releases. Jira's Portfolio allow to apply filters that can display a different granularity on what is shown, this means that it can be used at more levels in the company's hierarchy. With the kind of tools currently in use it's not possible to have such a smooth integration.

4.3.3 Client interaction

Although Service Desk would be the ideal software to allow the interaction with clients, Athonet has not yet thought of using it in such a way, but there is the possibility to do so in the future.

4.3.4 Internal documentations

Confluence substitutes the current multiple

4.3.5 The difference between these and other internal tool

HOW ATLASSIAN'S SOFTWARE DIFFER FROM Sharepoint, otrs, office 365

4.4 The Atlassian Community

Athonet has chosen wisely to buy the licenses Jira and Confluence over other tools and over creating their own internal solution, which would have meant reinventing the wheel.
Compared to other tools, Atlassian has built a large community around it's products, allowing people to ask and answer questions on a blog, request for new features, open tickets, view

the dates and changelog of next releases and report bugs. This allowed me to easily find not only the resolutions to some implementation problems I had during the installation, but tips on how to better configure the software as well.

Scrum means "Waterfall but we don't have time for analisys"
Kanban means "Scrum, but we don't have time for Sprint planning"
Agile means "We have no process but we do use Jira extensively"

@mikeveerman on Twitter

5 Projet implementation

This chapter is the core of this document and describes the way that this project has been implemented according to the planning described in Chapter 2.
It is structured in four main sections that go over the main time periods presented in the Piano di lavoro document.
Since the scope of the project was to create a demo environment to show the capabilities of Jira, Athonet has bough ten user licenses for Jira Software, ten for Service Desk and ten for Confluence.
All these software have three installation options:

- Cloud: on their infrastructure
- Server: download and install on a local network
- Data Center: download and install for a large infrastructure

Because of Athonet's policies about storing their data in the cloud, they opted for an on premise Server installation. This kind of installment requires a one time payment for the software while other subsequent fees will be for the support.

5.1 Learning phase

This phase corresponds to the first two weeks of the internship. As described in the work plan in this first period the main task was to understand what the tools are and what they can do.
At first I have started to search information about Jira and Confluence on Google. The first one I have started researching was Jira, and i found that the official documentation is very well organized.

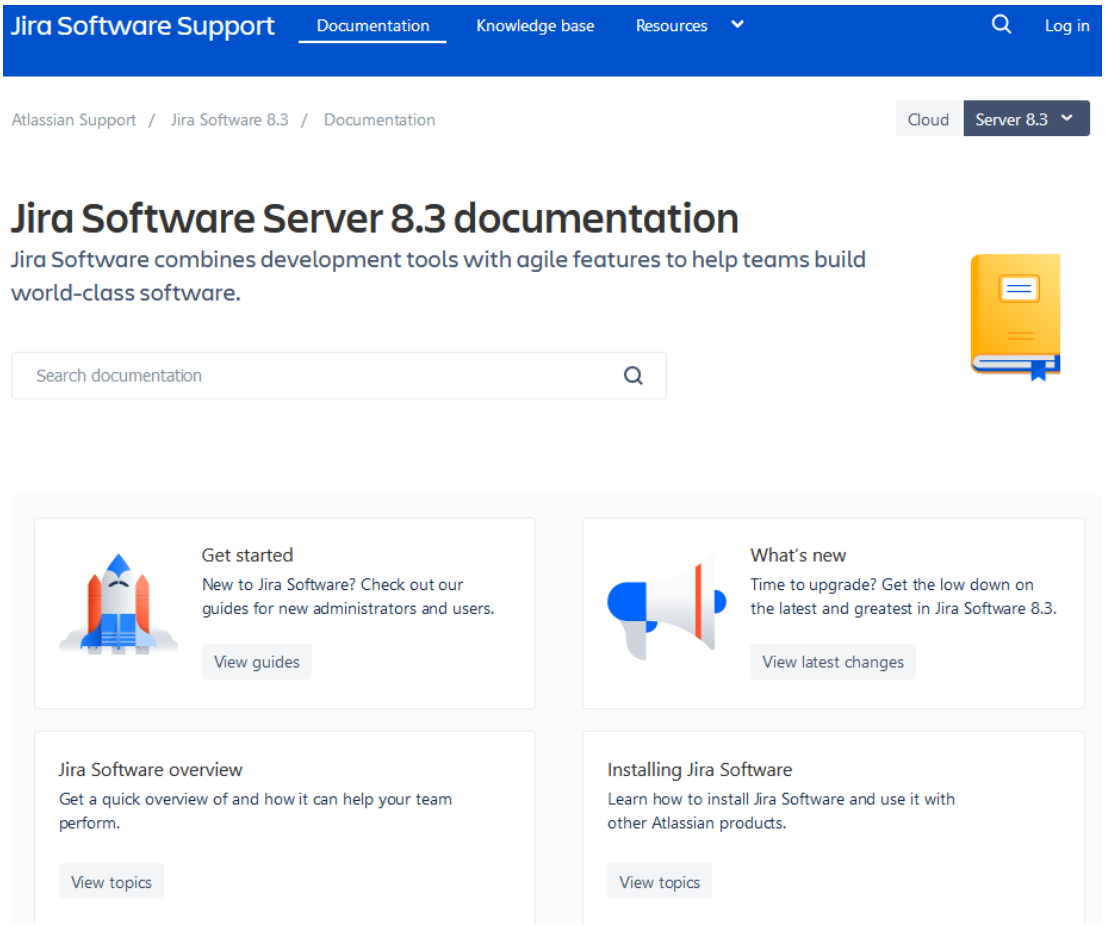

Figure 5.1: Screenshot of the Jira Documentation homepage from the Atlassian Support website

It is very easy to navigate because it's versioned for each release, major and minor, of the software, plus it contains links to related pages, including Confluence's documentation and Atlassian's blog. As said in ... both Jira and Confluence have a bug reporting and issue tracking section in their documentation.

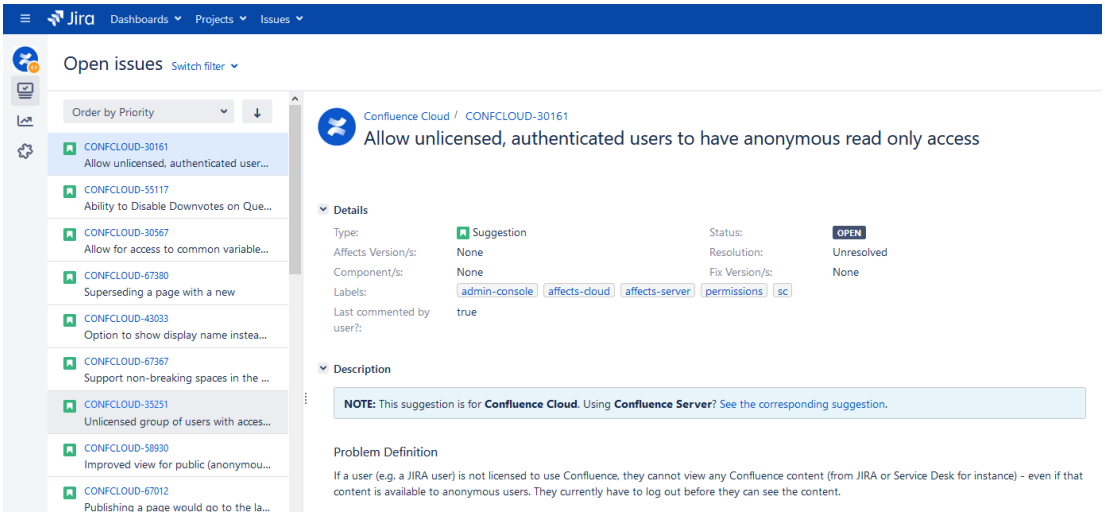

Figure 5.2: The issues related with Jira and Confluence are handled by a dedicated Jira Cloud instance

If a web page in the documentation is related to an issue, this is showed at the end of the page with its status and a link to its dedicated section.
The Confluence and Jira documentation are both written and hosted using Confluence, showing how powerful can be this tool for handling a wiki for such a complex software that has a large userbase.
Confluence's documentation is structured like Jira's, very easy to access and consult. Another bonus is that it is public and free to read, despite the software is not.
During the second half of the first week I went ahead and started configuring the environment. In order to install the Atlassian tools I was given a CentOS VM with 512GB of storage and 32GB of RAM, connected to a special testing network environment for internship students. To connect with the remote machine I used Remmina, a remote desktop client for Linux OSs.

Figure 5.3: The issues related with Jira and Confluence are handled by a dedicated Jira Cloud instance

5.2 Initial installation and configuration

Since the documentation I found was very exhaustive, I decided to anticipate the installation phase so that I could have a more hands on approach. I started to look on this more practical part from the second half of the second week and, although in the Gantt diagram it overlaps with the previous task, it's because it involves some study as well.
As for the previous phase, I started by installing Jira following the official documentation.

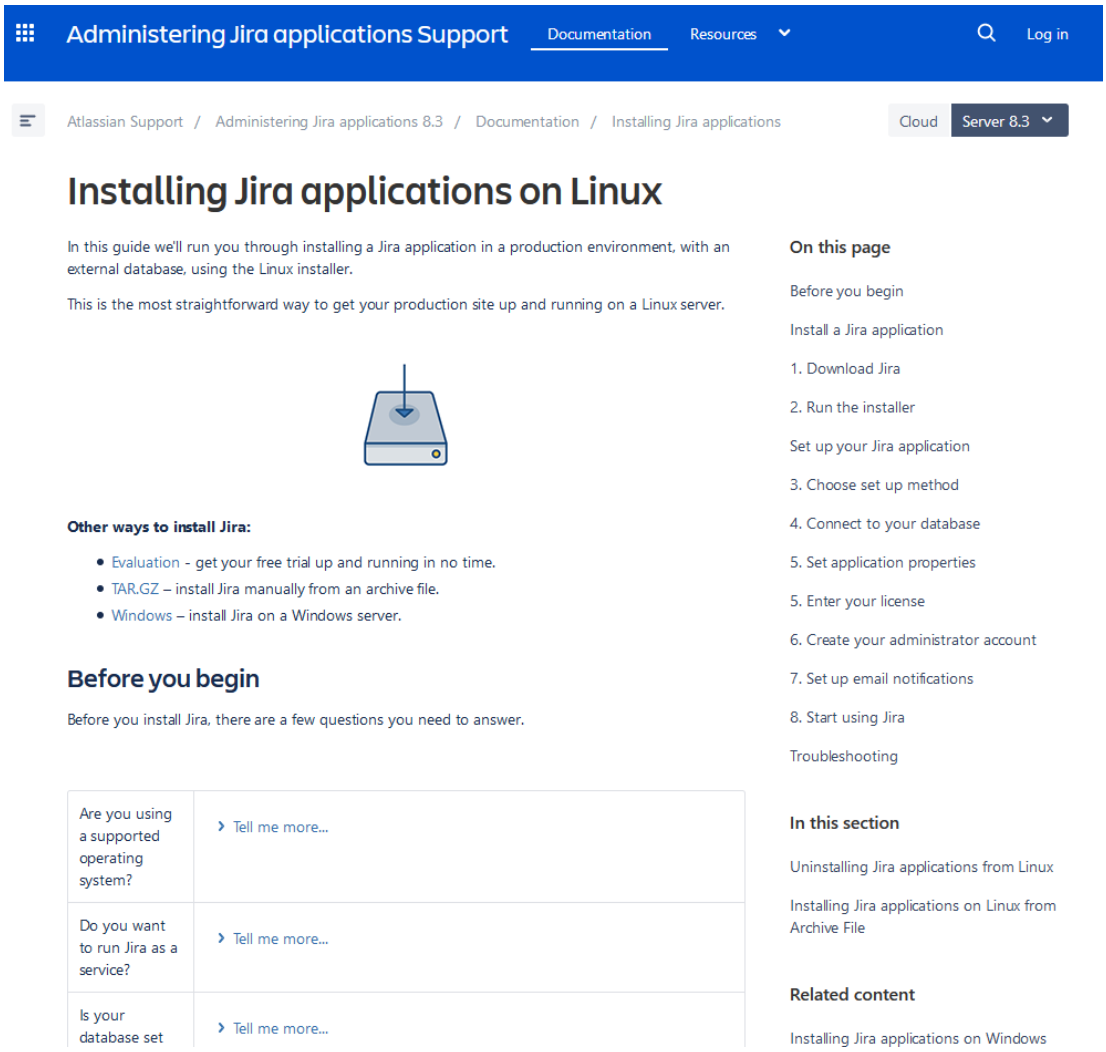

Figure 5.4: The issues related with Jira and Confluence are handled by a dedicated Jira Cloud instance

Jira needs a database in order to store all the information regarding projects, users, etc. For the first installation, used for testing purposes, the embedded H2 database was enough.

It's important to note that the documentation says the H2 database is not suitable for production environments. The first thing that I have done after the installation was getting acquainted with the interface and understanding how Jira's components interconnect with each other as I read in the documentation.
To do this I have created some mock projects and filled them with issues; it's here that I understood the concept of Board in Jira.
Boards and workflows are very tied in notions. Experimenting with workflows was one of the most important things to do, because these are fundamental in an issue tracking system's configuration and are strictly connected to Boards.

After understanding the fundamentals of this software and getting to know it's basic features, I went ahead and installed Portfolio. This plugin, as told in ... helps visualizing the issues on a roadmap, which was one of the most important requirements.
Installing Portfolio was easy, I just followed the instructions on the documentation to download the latest version of the plugin and embed it to Jira.

After installing this plugin and creating plans for my mock up projects, I have chosen to install Service Desk, to complete the configuration of my Jira instance. As earlier I have followed the documentation on the website. As described in ... this piece of software is used to communicate with the clients and having a portal from which a client can find information and request assistance with a product. The first thing I tested after installing this software was creating a help center and see from the point of view of a client how this can be able to open an issue and what type of issues he may have access to.

After I became familiar with Jira, I have installed Confluence, and like for the other software, I have used it's embedded database. Both software run on the same system, and their services are offered on port 8080 and 8090 respectively for Jira and Confluence.
During installation I connected them together and tested the environment by creating a knowledge base for a Service Desk project and a documentation space for a Software project.
Confluence was easier to get familiar with, so after creating some mock spaces related to the projects that were in Jira at the time, I moved on. To ease the integration between these tools, Atlassian allows to have a single users table stored on either Jira or Confluence.
This was shown during the installation of Confluence as an option to link it with Jira's instance was presented, I inserted the administrative credentials and told it to use Jira's users.
Jira and Confluence both use an Apache Tomcat web server. To run them both on the same domain I had issues regarding the login. Cookies for the users are stored for allowing them to access without inserting the password each time, but they didn't work correctly because the

users were automatically logged out after some minutes. This was because, when running on the same domain, even if on different ports, cookies from Jira and Confluence go in conflict. While seeking the Internet for answers, I found that many people new to these software gt into the same situation. To solve this issue all is necessary is to change the context path of the software's server.xml file that contains the configuration of the web-server. An example of changing the context path for this project is:

```
http://yourdomain.com:8080
```

becomes:

```
http://yourdomain.com:8080/jira
```

For Confluence, where the default port is 8090, the context path will become:

```
http://yourdomain.com:8090/confluence
```

At this point there was a change in the requirements that were given to me by the tutor. Contrary to what he said in the beginning, the IT department opted to use GitLab instead of BitBucket because the developers know it better and there is no billing for their usage tier. So it's free.

This meant that there would be one less tool that needed to be installed, but I had to understand how to link Jira's functionalities to those offered by GitLab.

Connecting these tools together means that a developer can interact with Jira's issues in a project by typing it's ID in the messages that he uses for commits, comments, merge requests and so on. Fortunately GitLab's documentation had a dedicated page with instruction examples and a guide that allowed me to configure both tools.

This functionality allows GitLab to interact with Jira's default workflows, which sometimes are too simple for some software projects.

Since the license for GitLab's hosted version is not Premium (or Silver for the online one) there is no interaction the other way around, from Jira to the repository. Because these tools are not strictly related, there is no 1:1 project mapping from GitLab to Jira, a commit message in the repository may reference multiple Jira issues from different projects, so it's the programmer's discretion to comment wisely. Later ... I will talk about installing a plugin that allows these tools to be better connected.

After understanding the potentiality of this connection I went on and set up a small demo that touched all the things that I have covered in the first two weeks of the internship. Since

my tutor wasn't available for a few days so I took the liberty to customize the environment by changing the colors and logos in the interface, putting Athonet's.

When he came back during the meeting I have taken with him, he said that he liked the work I have done and that it was time for more elaborate mock projects in order to present the tools' functionalities to other company figures.
After that I have deleted the mock projects from Jira and the related spaces in Confluence to ask the IT department for a snapshot of the VM I was working on.
This allowed me to have a baseline, a deliverable that I was able to use as a secure point in time in which I could go back if anything after went wrong.

5.3 First realistic mock projects and feedback

The fourth week of the internship I was ready to implement more realistic projects in Jira, connecting them to Confluence providing them with documentation and creating a link with GitLab. The objective was to finalize the working demo that could be shown to various members of Athonet for them to understand how these software can be used in their departments.
In Jira I have created the projects EPC and Dashboard, both Software, while in Confluence I have created the spaces EPC Documentation and Dashboard Documentation. Also in Jira I have created an Athonet Internal Wiki project, to show how Service Desk could be useful for sharing internal documents between employees.
For the first project I have implemented a more articulate workflow that resembles the realistic evolution of an issue inside the company and customized the menus that allow the creation of an issue.

Later these will be revised because in this stage I was not given the information about what an issue requires to have in Athonet's case.
As I continued working on the mock projects and creating issues, I noted down all the most important customization that an administrative user can use to set up the software, not only for Athonet's specific purposes but in general.
To make the project more realistic I have created three user groups, besides the default ones, to which I have assigned three users each:

- Management
- Verification

- Developers

Every group had various permissions; this allowed me to demonstrate how basic security rules work in these tools.
After telling the progress I have made to the tutor, he was able to set me a meeting with himself and the verification and developer managers. I then prepared some slides to explain some of the nomenclature in Jira, as explained in ..., and the various things that I have implemented.
The core of this meeting was a discussion between the managers focused on understanding how their ongoing projects could be implemented in such a way that the employees would not be forced into using a strict Agile methodology, but to accommodate and let them understand how these tools work without creating chaos.
There was a discussion on how applying a strict Agile methodology could impact their time division. Agile provides sprints that by definition last two weeks. These produce deliverables that bring added value to the project. Athonet's line of work though is focused on adding value to the product upon new releases that are scheduled differently by their purpose:

- Bug fix
- Minor release (for patching issues with the code or to improve performance)
- Major release (for new features)

Sprints would disrupt the routines of the developer team, the verification team and of management. In this case it would not be the tool that adapt to the company but vice versa the company would need to change in order to use the tool and would be counterproductive.
The developer manager stated that he wanted a way to measure productivity, so I showed him the reports section that is present for all kind of Jira projects.

Since he was not keen on the strict Agile interface with the backlog composed only by the issues that were meant to be completed in a two week period, he wanted me to show how the kanban board looked like.

He said that it would be better for the company's way of work since developers could program their work based on what has to be completed for the next release, so for a longer period of time.
This implied there would be a more particular workflow that had to be implemented and the managers were happy to see that it could be easily done, as for customizing the issue creation

and modification menu.
They were also happy about the Internal wiki project because it suggested there could be one single tool for both internal documentation regarding clients and general notes for employees (like for example a guide on how to install Docker without loosing internet connectivity).
After this meeting, I was put on standby because of the internal discussions that had to be done in order for the managers to understand how to adapt the workflow they had in Redmine for Jira and how to include Confluence in the mix. This was not considered when writing the piano di lavoro document, but to carry on with the work I decided to start writing the documentation about installing and configuring the software, aimed for the system administrators in the IT department. In the sixth week I showed it to the tutor and, after his approval, I started working on the user guide, which was written in the wiki Confluence space. In order for the users to understand how to use the tools, they had to start using them.

Later on, a new meeting was scheduled with the product owner, who wanted to understand better Portfolio and how the concept of product matches the project denomination in Jira.

Project and product sound similar and the two concepts are often confused with each other. A project is a temporary endeavour, with a clear definition of what needs to be delivered and by when, it has a beginning and end date. A product is designed to continually create value for customers by solving their problems.
For the product owner, one of the most important things was to see how releases are mapped inside the software, since Redmine did not offer such an intuitive interface and did not allow to see them on a timeline and applying custom filters. I showed him the progress I have made and he approved it, also he said that he wanted the credentials to start using it and see how to map the ongoing projects in Redmine to new ones in Jira or how to migrate them.
After using it for a couple of days and talking with the other managers, I was given a feedback on how Athonet would use these tools. Since they were not sure letting users access Service Desk directly, an employee would create an issue that would represent the customer request through service desk. The customer request could be a new feature, a bug report, require support, etc. This would then be picked up by a team that would bring it inside a Business type of project by creating a linked issue. This project would have a particular workflow and contains the more high lever tasks like documentation, meeting notes, feature statuses, etc. These shared notes and documents would be stored in Confluence, where there would be links to them.
After they have been approved for production another team would create a linked issue in-

side the software project. This would most likely be divided in many sub-issues by the team leader and these would be visible by the developers. Each developer could then pick an issue to work on and handle it as he wants by, for example, dividing it in smaller sub-issues.
The next week I had another meeting with more company figures: a senior developer, the business strategy manage, the managers that I have talked to in previous meetings, the CTO and my tutor. As in the other meetings I have explained the various software that I have installed, the purpose for each one of them and the evolution of an issue.
Again, to have a milestone, I have requested a snapshot of the machine by the IT department. Even with the requirements changes I was on schedule with the Gantt diagram and the other planning.

5.4 Transitioning into Production and final feedback

Although in the first planning this was a longer period of time, this last phase was shortened to a week. This was due because of the many internal discussions that took place for the software to be approved and for the various managers of the departments to figure out how to introduce it to their subordinates.
After the last meeting described in the previous paragraph, I have created new users for the managers that participated and gave them the credentials so that they could see the interface and interact with the mock projects that remained there.
The tutor had credentials as well and he was very enthusiast of how the demo performed and the potentials of the software, although this had to be well understood by the other managers and members of the company as well.
Although the last snapshot of the VM that was made contained a working infrastructure ready to be moved in a production environment, the IT department opted to wait until they understood how to properly secure the server that was going to host it. The most important part would be creating a secure VPN tunnel that would allow outside employees that are working from other locations to connect as they would be inside the building. This method must ensure thought that there would be no possibility for any other to get, or even hack, inside the company's network and access their products' source code. Besides this, the other managers where very glad to hear that the tools have been approved and where happy about the demonstration.
Because this, I was not able to migrate any data from Redmine's instance to Jira, since I was not granted permission to access the company's private data about the source code and even

if the migration would have been successful it would have not been used when Jira and Confluence were moved into production. The CTO of the company and PriMo's product owner were interested in the possibility of using a real time roadmap to keep track of the work done while not going as in deep as seeing the tasks assigned to each developer.

5.5 How are these tools being used

Athonet does not want to pursue being an Agile software developing company in the near future, but it wants the tools that allow to organize their projects such that if one day their strategy varies, then the software would follow along. What they really want adopt now is a hybrid between Scrum and Kanban: Scrumban.
As in Kanban visualizing workflow, limiting work in process (WIP), and measuring productivity are prioritized. Scrumban removes the practice of limiting WIP by time (i.e. velocity and Sprint boundary) and replaces it by limiting WIP for each stage. It's based on having a continuous flow of work, which is what Athonet has for their products.

6 Conclusions

6.1 Improvement and future implementations

6.2 Final Gantt diagram

MOVE IMAGE IN APPENDIX

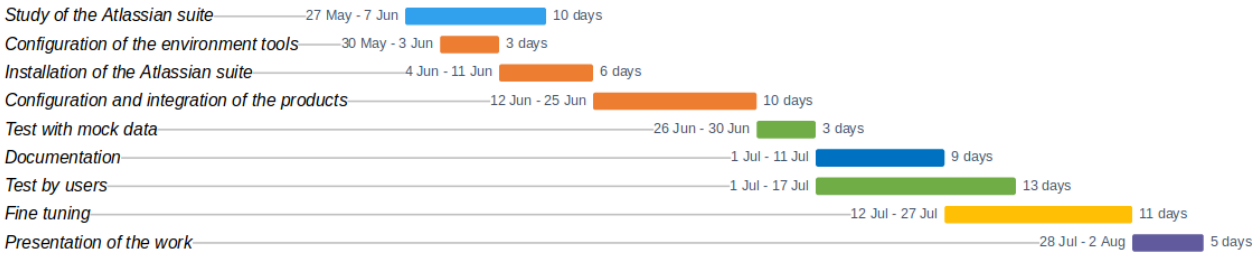

Figure 6.1: Dilbert on Agile Programming

6.3 Objectives achievement

6.4 What I have learned

6.5 Personal considerations

A

Appendix A: Gantt Diagrams

This Appendix contains the ...

A.1 GANTT 1

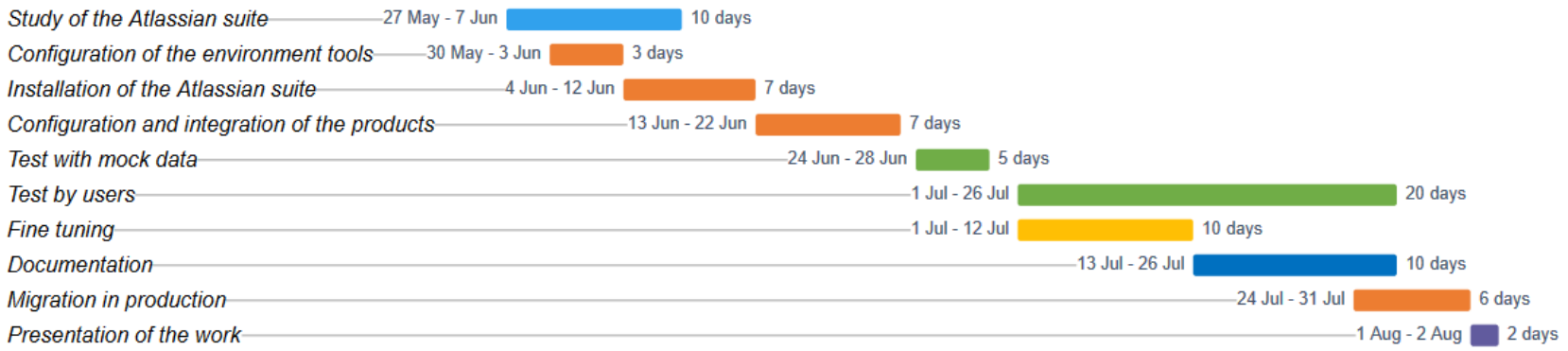

Figure A.1: Gantt diagram contained in the "*Piano di Lavoro*" document

A.2 GANTT 1

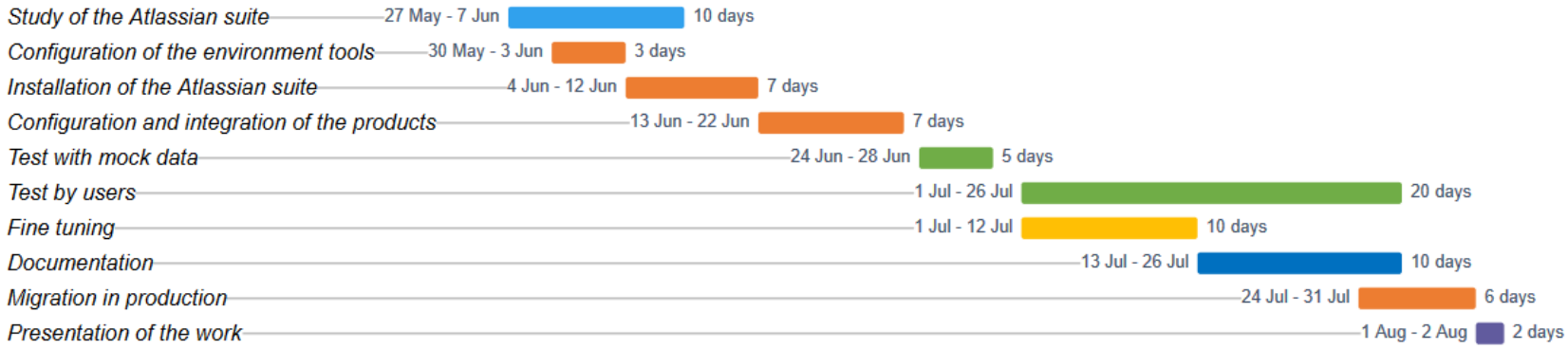

Figure A.2: Gantt diagram contained in the "*Piano di Lavoro*" document

B

Appendix B: Project requirements

ADD COMPLETE TABLE WITH THE PROJECT REQUIREMENTS

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

References

[1] Dilbert official website. [Online]. Available: https://dilbert.com/

[2] Athonet official website. [Online]. Available: https://athonet.com/

[3] Ericsson. [Online]. Available: https://www.ericsson.com/en

[4] Gianluca, inventore di reti low cost ora entra alla corte di facebook. [Online]. Available: https://www.ilgazzettino.it/vicenza_bassano/bassano/telecomunicazioni_gianluca_verin_athonet_zuckerberg_facebook_bassano_roma_karim_elmalki-1936961.html

[5] [Online]. Available: http://www.ilgiornale.it/news/interni/genio-torna-svezia-soffrire-nella-sua-italia-912407.html

[6] [Online]. Available: https://www.millionaire.it/athonet-la-startup-italiana-che-trionfa-al-mobile-world-congress/#!

[7] [Online]. Available: https://www.redmine.org/

[8] [Online]. Available: https://products.office.com/en-us/sharepoint/collaboration

[9] [Online]. Available: https://products.office.com/en-us/business/task-management-software

[10] [Online]. Available: https://about.gitlab.com/

[11] Software life cycle. [Online]. Available: https://ase.in.tum.de/paid.globalse.org/paid1/courseDocs/Readings/SoftwareLifeCycle030398.pdf

[12] [Online]. Available: https://www.iso.org/standard/53815.html

[13] [Online]. Available: https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development

[14] [Online]. Available: https://www.productplan.com/glossary/agile-manifesto/

[15] [Online]. Available: https://agilemanifesto.org/

[16] [Online]. Available: ttps://medium.com/hygger-io/4-values-of-the-agile-manifesto-and-12-agile-principles-easily-explained-84cd429f69f

[17] [Online]. Available: https://explore.versionone.com/state-of-agile

[18] [Online]. Available: http://www.ambysoft.com/essays/agileRoles.html

[19] [Online]. Available: https://www.inc.com/adam-fridman/the-massive-downside-of-agile-software-development.html

[20] [Online]. Available: https://www.theserverside.com/discussions/thread/24701.html

Glossary

BROADBAND NETWORKING

: a fruit. 51

CORE NETWORK

: a fruit. 51

LOW LATENCY COMMUNICATION

: a fruit. 51

LTE

: a fruit. 51

MISSION CRITICAL

: a fruit. 51

REPOSITORY

: a fruit. 51

ROADMAP

: a fruit. 51