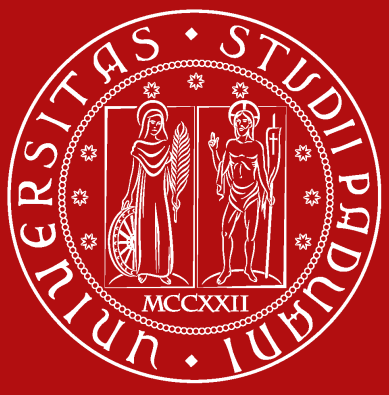

DIPARTIMENTO
MATEMATICA

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: 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
Armir Bujari

Candidate
Ciprian Stefan Voinea
Student ID
1143057

Accademic Year 2018 - 2019

This is a dedication

Abstract

la crescita in base al numero di dipendenti necessità di avere un tool complesso e sofisticato per gestione della parte di sviluppo sw non è il solito gestionale, ma sono tool specifici che considerano i trend a livello di sviluppo

raccontare risultato ottenuto dello stage il lavoro di questa tesi è stato ... tool più conosciuto dal mercato ... installare / config ... ottenere approvazione da management .. migrazione dei sistemi in uso nel nuovo gestionale

Sommario

la crescita in base al numero di dipendenti necessità di avere un tool complesso e sofisticato per gestione della parte di sviluppo sw non è il solito gestionale, ma sono tool specifici che considerano i trend a livello di sviluppo

raccontare risultato ottenuto dello stage il lavoro di questa tesi è stato ... tool più conosciuto dal mercato ... installare / config ... ottenere approvazione da management .. migrazione dei sistemi in uso nel nuovo gestionale

Contents

Listing of figures

Listing of tables

1 Introduction

introduzione a significato di way of working (in SW) cercare articoli da blog per spiegare martin fowler

fare tracking delle isssue / bug è diventato difficile / complesso molti tool open source che fanno anche documentazione e code review (altri aspetti)

l'evoluzione negli ultimi anni nelle aziende IT necessità di organizzazione delle azienda e la necessità di avere una gerarchia (o simil gerarchia)

ho sperimentato questo approccio in athonet, mostrata interessata all'utilizzo di un gestionale sw di tipo agile per la gestione dei processi di sviluppo sw interni

I have sperimented ...

1.1 Premise

This document represents 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. It satirically represents the problems that can be present in a small or big company of software development.

1.2 The company

Athonet is a telecommunication company headquartered nearby Vicenza. It was born from the idea that broadband networking should be easier to access and more available for people in rural areas and for 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 was already developed by the CEO and CTO that were working alongside in Ericson at that time.

Figure 1.1: Athonet's logo

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

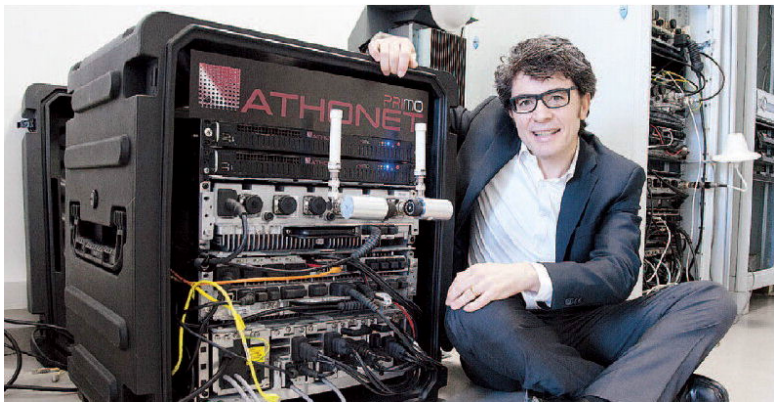

Figure 1.2: The CTO, *Gianluca Verin*, with Athonet's main product ***PriMo***

They showed how this product performs in an emergency situation on the field in Emilia Romagna, when in 2012, after the disastrous earthquake has destroyed all the communication lines Athonet has reestablished communications. Athonet arrived with PriMo and installed it 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, along other companies, 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. 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 company may still be small but has much to offer, considering that some of it's competitors are giants like Nokia and Ericson. Athonet's passion and dedication are demonstrated through the awards and prizes that has won, for example this year, 2019, has won a total of 4 Global Mobile Awards including CTO's choice at the GSMA in Barcellona.

1.3 The project

I have chosen to do my internship at Athonet because the project, although may seem simple, is quite interesting. 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 document 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 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

2

The internship project

This chapter contains the results of the discussions and planification that have been made prior to the beginning of the internship. These contents are described in more detail in the Piano di Lavoro, a document that contains an estimation of resources for each task that composes the project.

At the beginning of May I have met the tutor in order to understand what the company wants and make a first time planification.

In here is also described how a resolution for the problem was first thought.

2.1 The company's needs

As said earlier Athonet is a growing company. At the time of writing it has nearly 40 employees and it is expected to hire new people for open positions. 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 as an Issue Tracking System
- SharePoint as a collaborative platform
- Planner for visualizing project roadmaps
- GitLab as a repository

Not all of these have been created with the idea of being used together. There are plug-ins that connect them, but these don't allow much customization and since are made from multiple developers, not always are compatible when a new software release is issued. 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.

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

When discussing with the tutor he as explained me the problem step by step and we have set some requirements in order to give them importance and priority. The final objective was to create a suitable environment that would work and 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 importance:

- 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 a number.

Here a list of the requirements, as presented in the work plan document:

2.3 Approaching the problem

How I thought I could approach the problem. Write that I have thought of use chases and associate them to the requirement?

2.4 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 ten weeks, which is a little more than two months, and the project has been spread such that I could have time to understand the tools and use them so that I could 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.

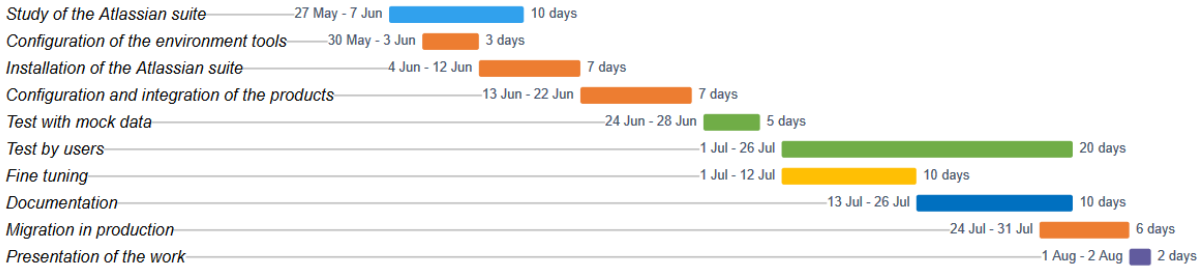

Figure 2.1: Gantt diagram contained in the "*Piano di Lavoro*" document

I started to create a temporal sequence only after I well understood the requirements and got an idea about how Atlassian's software works. To create a good plan I have started from understanding what was the final objective and asking myself the question of how can I achieve it.

I have coarse grained planned four main time periods:

- Learning
- Implementing
- Testing
- Feedback

Each of these contain smaller tasks that involve studying the tools, installing them and configuring them. As discussed with the tutor, meetings had to be considered so that I could explain the progress to him and to other company figures that would eventually use the software.

As milestones / baselines 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.

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 and 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 of the software. As activities are assigned to different persons, 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 dependencies among other activities 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.

There are many SDLCs like Waterfall, Prototyping, Iterative and Incremental Develop-

ment, Spiral Development, Rapid Application Development, and Extreme Programming. (XP).

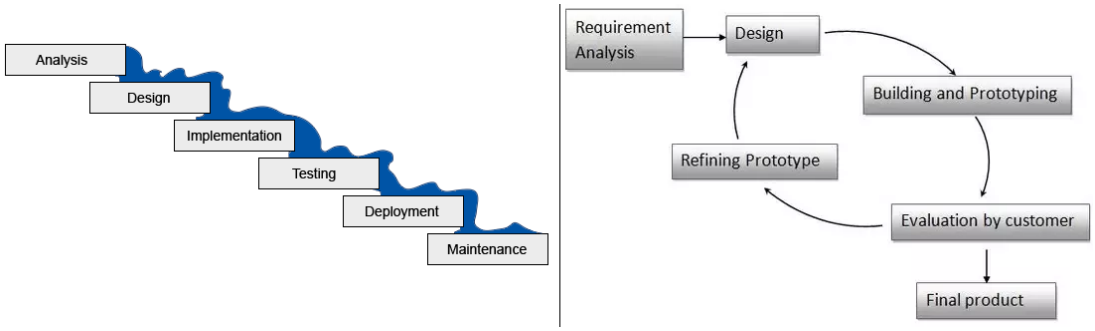

Figure 3.1: The Waterfall and Prototype SDLC models

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.

The complexity and slowness in producing a concrete product in older SDLCs brought the need for a faster and more communicative model.

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 Why 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.

SDLC models can be of two types:

- Iterative: (like ..)
- Incremental: (like ..)

Two characteristic models used before Agile where the Sequential model (or Waterfall) and the Incremental model.

The pros / cons of waterfall were ... The pros / cons of x where ...

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.

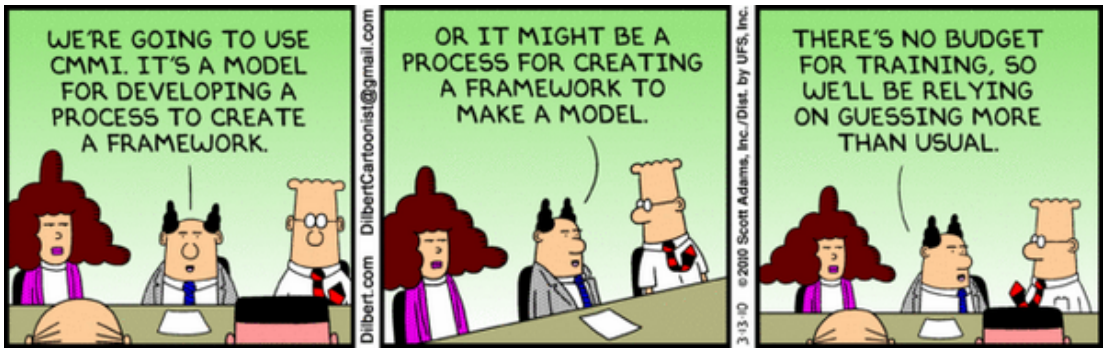

Figure 3.2: *Redmine*'s logo

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.

The four values 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.

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 from VersionOne's team are:

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

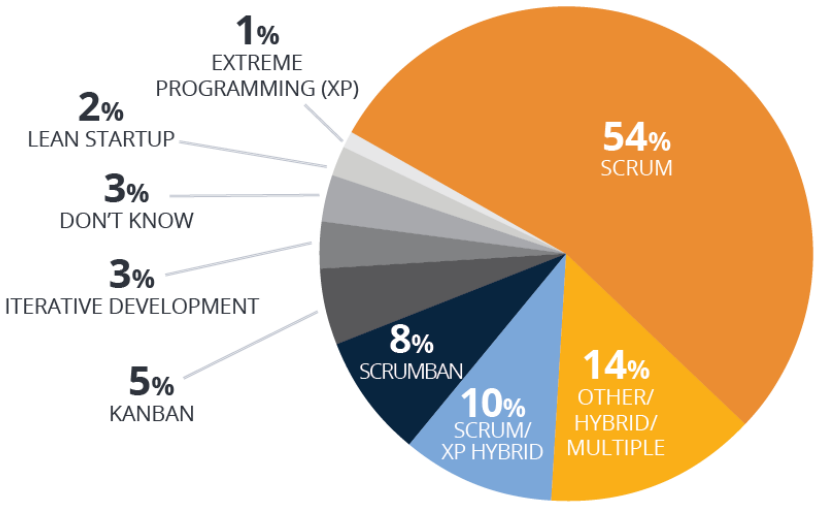

Figure 3.3: The Waterfall and Prototype SDLC models

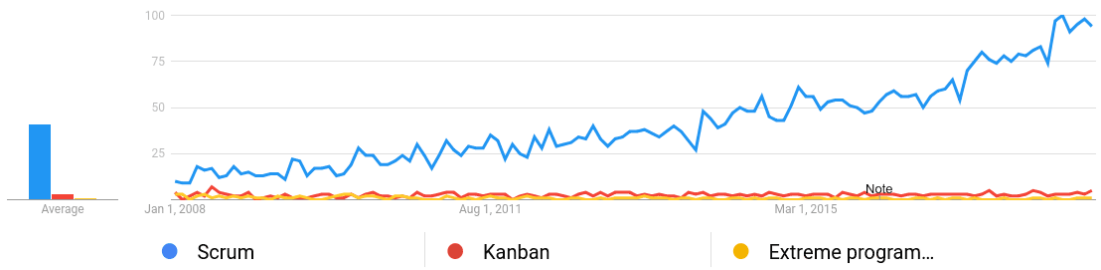

Figure 3.4: The Waterfall and Prototype SDLC models

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.

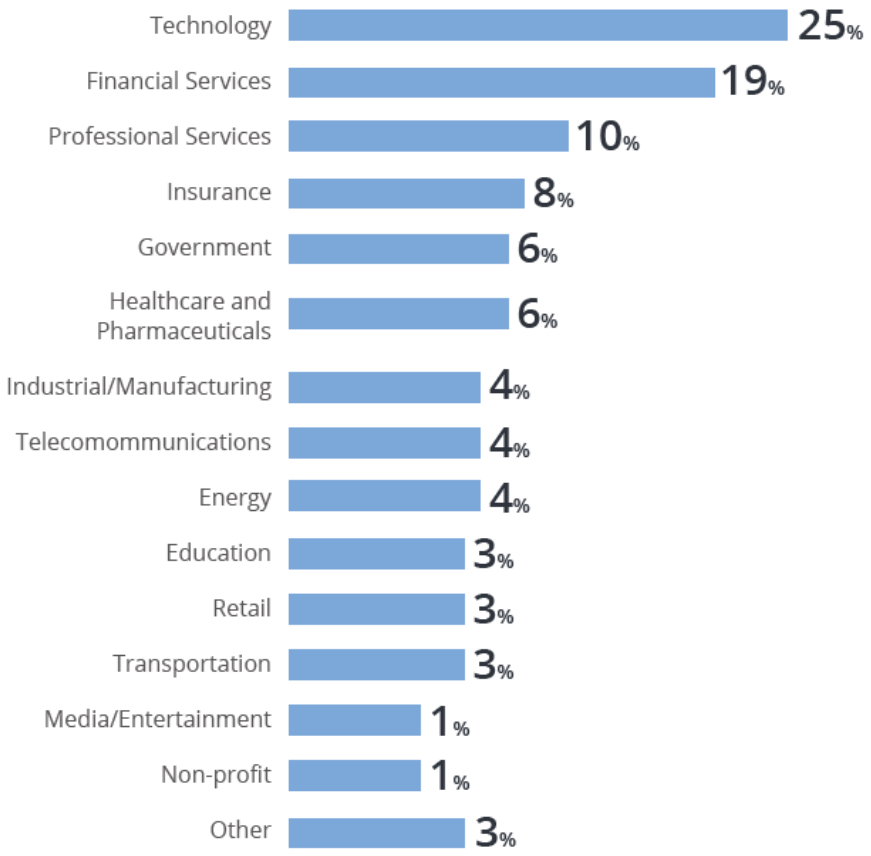

Figure 3.5: The Waterfall and Prototype SDLC models

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:

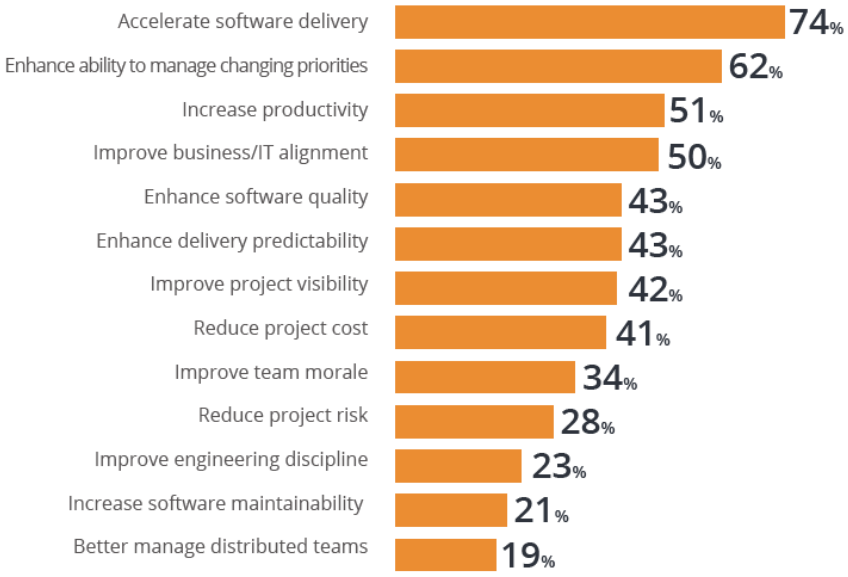

Figure 3.6: The Waterfall and Prototype SDLC models

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

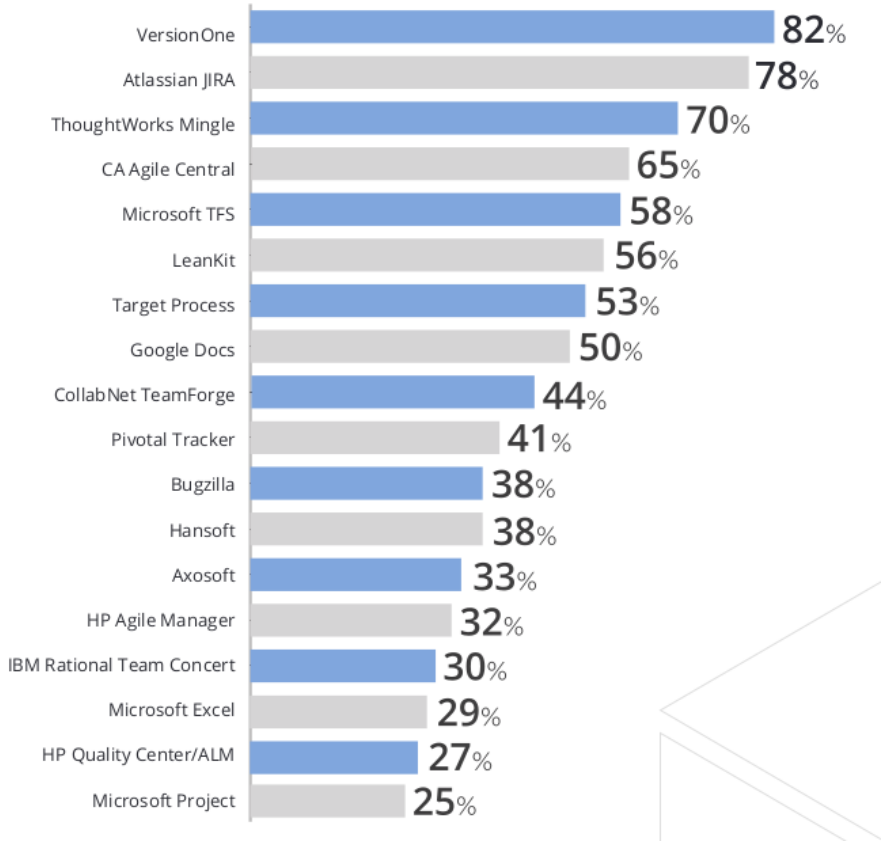

Figure 3.7: The Waterfall and Prototype SDLC models

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

Figure 3.8: The Waterfall and Prototype SDLC models

3.4 Agile in practice

Briefly describe how some companies use Agile: Spotify / Netflix

3.5 The roles in Agile

Briefly describe the most important roles in Agile

3.6 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 of Agile:

1. Less predictability
2. More time and commitment
3. Greater demands on developers and clients
4. Lack of necessary documentation
5. Project easily falls off track

Briefly explain the points

3.7 What Agile variant will Athonet use

As many small companies, Athonet will not strictly use one Agile implementation. This is due also 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 search for the available software development 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 3.9: 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. These are proprietary tools are 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*".

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.

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 will Athonet use them - CAMBIARE TITOLO

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

When trying to adapt such versatile tools it can become difficult to understand if the company must adapt itself to the

These tools are complex, it is necessary to understand the scenario they can be used in and how to transition from the old 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 xxx 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

Sharepoint, otrs, office 365 – POSSIBILE CITARLI PRIMA QUESTI TOOL

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.

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

5 Projet implementation

This chapter is the core of this document and describes the way that this project has been implemented according to the work plan described in Chapter 2.

It is structured in four main sections, each representing a time period:

1. Learning stage - RIVEDERE NOME
2. Implementation
3. Testing
4. Feedback

5.1 Learning stage - CAMBIARE NOME

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. At first I have started to search information about Jira and Confluence on Google. The first tools I have started researching was Jira, and the official documentation is very well organized.

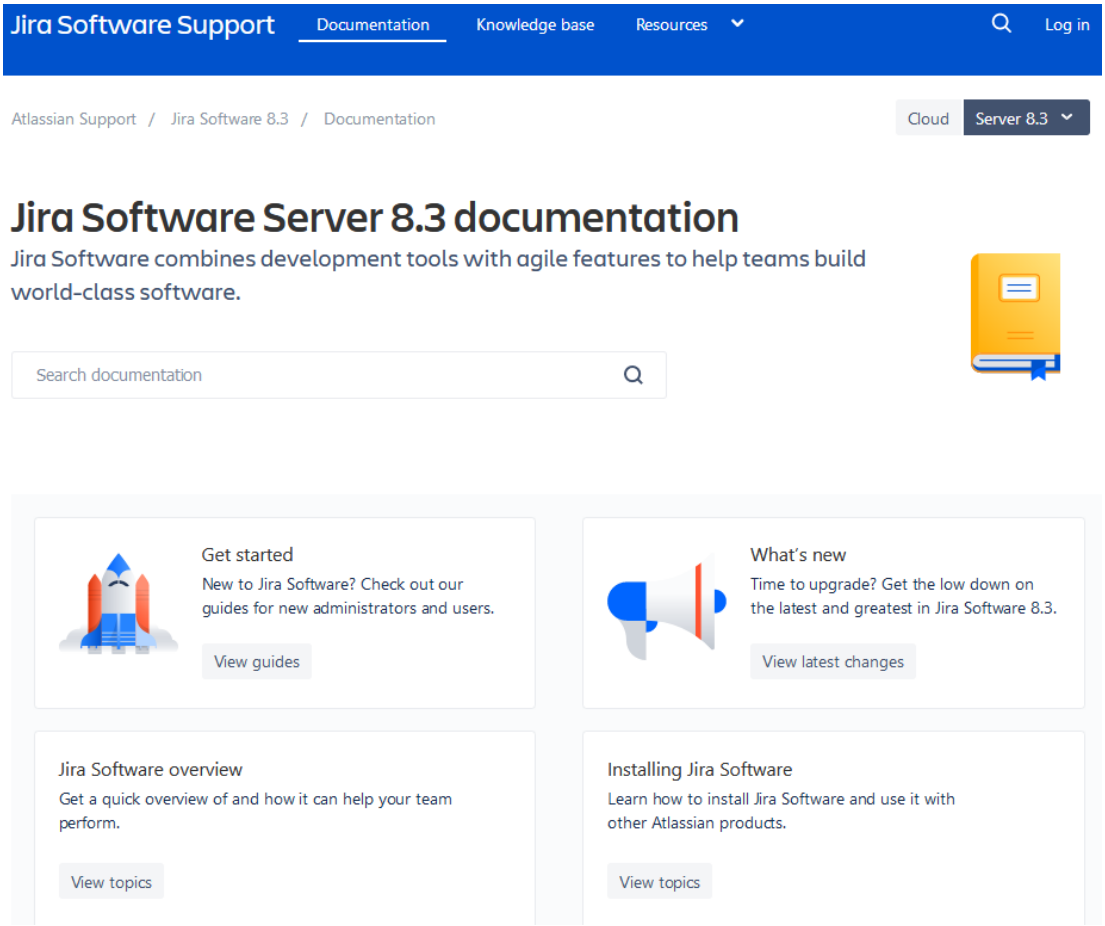

Figure 5.1: Screenshot of the Jira Documentation homepage from the Atlassian Support website

This documentation is very easy to navigate because it versioned for each release, major and minor, of the software, plus it contains links to related pages. If there is a reference about a Confluence webpage, this is added as a link. Vice versa on the Confluence documentation, if there is a Jira reference, it links to the latter's documentation. Both Jira and Confluence have a bug reporting and issue tracking section in their documentation.

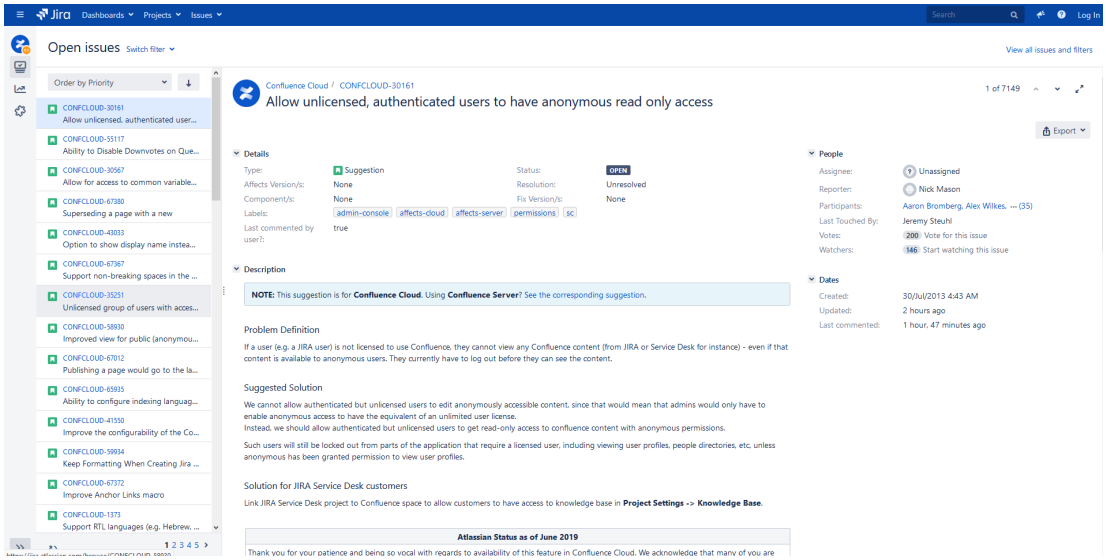

Figure 5.2: The issues related with Jira and Confluence are handled by a dedicated Jira Cloud instance

If a webpage in the documentation is related to an issue, the latter 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 consult, despite the software is not. This may seem like an obvious choice but not all vendors do it: RedHat for example let's you consult their documentation only if you log into the website.

5.2 Initial installation and configuration

During the second week, while studying the documentation I went ahead and started configuring the software. In order to install the Atlassian tools I was given a CentOS VM with 512GB of storage and 32GB of RAM.

To connect with the remote machine, which was in a controlled testing environment, I used Remmina, a remote desktop client for Linux operating systems. This allowed me to easily connect to the machine to install software or to troubleshoot it in case of a failure. As for the previous phase, the first software I installed was Jira, by following the official documentation.

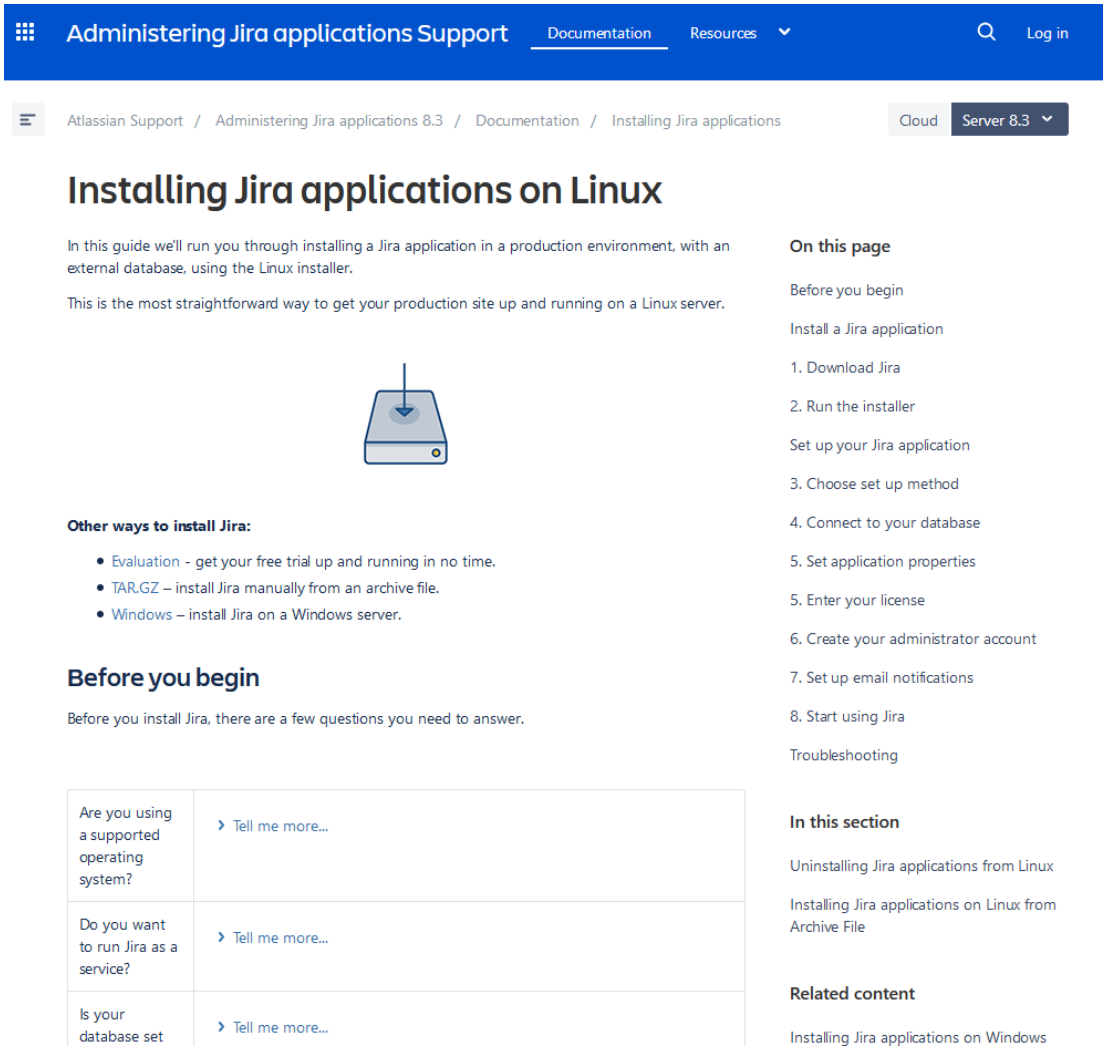

Figure 5.3: The issues related with Jira and Confluence are handled by a dedicated Jira Cloud instance

As told in ...aggiungere riferimento... Jira requires a database to work properly. For the first installation, which was made for testing purposes, the embedded H2 database was enough.

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. To do this I have created some mock projects that I filled with issues, it is here that I understood the concept of Board in Jira. 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 the concept of Board.

After understanding the fundamentals of this software and getting to know it's basic fea-

tures, I went ahead and installed Portfolio.

This plugin, as told in ...aggiungere riferimento... 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 install the latest version of the plugin.

After installing Portfolio and creating plans for my mock up projects, I have chosen to install Service Desk, to complete the configuration of the Jira instance. As earlier I have followed the documentation on Jira's website. As described in ...aggiungere riferimento... 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 portal 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.

Not long after I have installed Confluence, and like Jira, 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. Soon after I connected the software together and I tested it out 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.

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

This meant that there would be a tool less 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 that allowed me to configure both tools.

This functionality though allows GitLab to interact with Jira's default workflows. If the licence for GitLab's hosted version is not Premium (or Silver for the online one) there is no interaction from Jira to GitLab.

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.

Later ...aggiungere riferimento... 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.

During the meeting I have taken with the tutor, 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 milestone / 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 have a 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 projects, 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 did not have 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; that allowed me to demonstrate how basic security works in these tools.

As I told the progress that I have made to the tutor, he was able to set a meeting with him, the verification manager and the developer manager.

For this I prepared some slides to explain some of the nomenclature in Jira and the various things that I have implemented.

The core of this meeting was a discussion between the managers that verted on understanding how the various projects could be implemented in such a way that the employees would not be forced to use a strict Agile methodology, but to accomodate them and let them understand how these tools work.

This also meant that there would be a custom 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 meant that 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 plan of work document, but to carry on with the work I decided to start writing the documentation about installing and configuring the software, aimed for the administrators.

In the sixth week, after this was done 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. This meant that in order for the users to understand how to use the tools, they had to start using the tools and search in them the ”how to”s.

Later on a new meeting was scheduled with the product owner, who wanted to understand better how Portfolio works and how the concept of product matches the project denomination in Jira. For him one of the most important things was to see how releases are

mapped inside the software, since Redmine did not offer an intuitive interface and did not allow to see them on a timeline and applying 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.

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.

Snapshot of the machine.

———————————————————————————————————

5.4 Transitioning to production

This phase corresponds to ... in the work plan

After the approval for using the tools by other departments (R&D) it's time to transition it / move it to production

5.4.1 Migrating data from Redmine

tool automatico di migrazione collegamento con redmine, lo fa in maniera automatica e se va male? c'è sempre lo snapshot

5.4.2 First non mock projects

5.4.3 Fine tuning of the final product

interazione con le persone in base alle necessità degli utenti e di come lo usano faccio minime modifiche in produzione miglioramento della documentazione

5.4.4 How are these tools being used

è veramente agile? è un dialetto? è un misto? perchè athonet lo sta usando in questo modo?

5.5 Final feedback and what else could be implemented in the future

This phase corresponds to ... in the work plan

5.5.1 Final feedback from the users

feedback da parte del tutor

feedback da responsabile strategia aziendale (gl) feedback da responsabile del prodotto (aka product ownner / hesham) feedback da responsabile sviluppo + testing

feedback da parte di tutti gli utenti

5.5.2 What Athonet plans to do with these new tools

arrivare ad utilizzare agile in maniera rigida? continuare a fare misto?

6 Conclusions

6.1 Improvement and future implementations

Now that all the work has been explained, let's see how it turned out and what can be done to improve it in the future

6.2 Final Gantt diagram

come si discosta da quello iniziale
cosa ha causato questo discostamento
ho pianificato male

6.3 Objectives achievement

6.4 What I have learned

principalmente il way of working aziendale
come funzionano questi tool
il loro scopo di "ordinare" un'azienda

6.5 Personal considerations

in un'azienda in crescita è molto utile darsi dei paletti
in athonet funzionerà una cosa del genere? software monolitico / complesso (service oriented architecture)
sta funzionando questo tool adesso per il breve tempo che l'ho visto io in produzione? valore aggiunto all'azienda, cosa vede il cliente

A

Appendix A: Piano di lavoro

Aggiungere schermate del piano di lavoro

B

Appendix B: The Agile manifesto