CONTINUOUS INTEGRATION METHODOLOGY IMPLEMENTATION AND COMPARISON

Master's degree in Computer Engineering - Mobile Computing

Gabriel Dario Chanchay Tituaña

Leiria, September of 2021

CONTINUOUS INTEGRATION METHODOLOGY IMPLEMENTATION AND COMPARISON

Master's degree in Computer Engineering - Mobile Computing

Gabriel Dario Chanchay Tituaña

Project Report under the supervision of Professor Catarina Isabel Ferreira Viveiros Tavares dos Reis, PhD. and Professor Marisa da Silva Maximiano, PhD.

Leiria, September of 2021

Originality and Copyright

This project report is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Dedication

I dedicate this work to my family; my mother, father, and sister, specially to my parents that have been supporting me always.

I also dedicated to God that gave me wisdom to develop and implement this project.

Acknowledgments

I thank Saviasoft company for letting me implement and gather all information that I needed in order to accomplish this work. I thank the CEOs and my coworkers of the company without them this project could not be possible.

Abstract

In order to know the impact and results of a Continuous Integration (CI) Methodology an implementation of CI was made on a software company. Saviasoft is a company that produces custom software. Implementing CI over an existing agile methodology that has not been changed for a while, is not easy, mainly because the developers are unaware of the benefits of using CI tools. In order to accomplish the objective of implementing a CI methodology, developers should have enough knowledge about CI concepts and know how to use the adequate tools properly, and, how to implement them on their projects.

The aims of the project are to improve the quality of the software that Saviasoft produces; analyze the impact of the CI methodology implementation; and improve the reliability of the software that is developed, by having a better way of testing. Jenkins is the tool that will serve as the base for the CI methodology implementation.

The project implementation has the following steps: (1) Analysis of the actual situation of the company; (2) Selection of specific metrics to measure before implementing CI; (3) Implementation of CI over the existing agile methodology; and (4) Measure the same metrics after the CI implementation and compare the results.

The results of the CI implementation were the expected: currently, the company produces software with an enhanced quality.

The CI implementation mainly improved the software quality, test phase, and deployment phase. In the future, Saviasoft will propose CI courses to the clients that have in-house developers.

Keywords: Continuous Integration Implementation, Quality Assurance, Continuous Integration Comparison, Impact of Continuous Integration, Measure Continuous Integration.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

CI	Continuous Integration
CD	Continuous Deployment
CDE	Continuous Delivery
ESTG	School of Technology and Management
NFR	Nonfunctional requirements
NFT	Non-Functional Test
QoS	Quality of Software
RAD	Rapid Application Development
XP	Xtreme Programming

1. Introduction

With the increasing competition in software market, organizations pay significant attention and allocate resources to develop and deliver high-quality software at much accelerated pace [1]. Continuous Integration (CI), Continuous Delivery (CDE), and Continuous Deployment (CD), called continuous practices for this study, are some of the practices aimed at helping organizations to accelerate their development and delivery of software features without compromising quality. Whilst CI advocates integrating work-in-progress multiple times per day, CDE and CD are about the ability to quickly and reliably release value to customers by bringing automation support as much as possible [2].

The work proposed aims to implement a methodology of continuous integration on a software developer enterprise, comparing the results, inputs, outputs, software features, before and after the implementation of the methodology. One of the first features that was done with this thesis was the SonarQube [3] Implementation.

The case study will be conducted at Saviasoft (Ecuador). The company uses agile methodologies and thesis will implement CI on these methodologies. The actual process of the company just achieves few goals of CI. The goal also is to improve this process adding new tools and features to this process.

The implementation of this project in the company is important because it represents a big step that is the implementation of a continuous integration process. They realize that the actual process that they have implies that the tests of the software are at the end of the development phase. In order to get the information needed to accomplish this work, we will compare results of specific metrics measured before and after the implementation.

The implementation of CI will take more time on the early stage of the implementation, nevertheless when the process of CI is clear, it is expected that the time on each phase of the development will decrease.

1.1. Objetives

The principal objective of this project aims to have and implement CI and CD on the enterprise in order to have better software. The actual situation of the company regarding its agile methodology will be analyzed. In order to get results before and after CI certain software measures will be analyzed. On each software development phase of the company will be implemented CI, after CI implementation there will be an analysis in order to know how the implementation affected, changed, improved the agile methodology and software of the company.

- Analyze the actual situation of the company regarding Continuous Integration.
- Analyze and select measures to get result before CI with a case study.
- Implement CI on each phase of the actual software development methodology.
- Analyze the impact after CI with a case study.

1.2. Document Content

- Chapter 2 - This chapter introduces the DevOps approach on this dissertation, an introduction of this topic is needed because DevOps involves an important part on this dissertation.
- Chapter 3 - This chapter describes the methodology used to implement this dissertation, the courses dictated to introduce the CI tools, and the state of the company describing the agile methodology that the company uses.
- Chapter 4 - This chapter shows how CI and CD was implemented on the company using a case study to describe each tool that was implemented.
- Chapter 5 - This chapter show the results of the CI implementation, the comparison between the input, metrics of software before and after the implementation.

2. DevOps

The continuous progress of the software industry has been affecting all software development phases. Before 2001, the software industry used traditional software development processes such as: the classical Waterfall model [4], the Iterative Waterfall model, the spiral model [5], or the RAD model [6]. Due to the criticisms and the high ratio of software failures that used these traditional models, a change in the software process development emerged in 1999 [7]. This evolution started to encourage lightweight processes and a broader approach to ensure high quality software development.

This is the reason why Agile methods [8] were born, they have transformed the way software is developed, emphasizing active client involvement – the client and his feedback with each delivery of the project , and an evolutionary delivery of products – a faster and a better way of deliver the software (Figure 1). An important issue of agile development is that it focusses on "feedback and change" [9]. The feedback of the client is important because it gives to the developers and the enterprises some evidence and the guidance that are important to have a satisfied and happy client.

In 2001, the original contributors of this evolution met and tried to identify the areas that these existing software methodologies had in common. Focusing on this common ground, led to the "Agile Manifesto" [10].

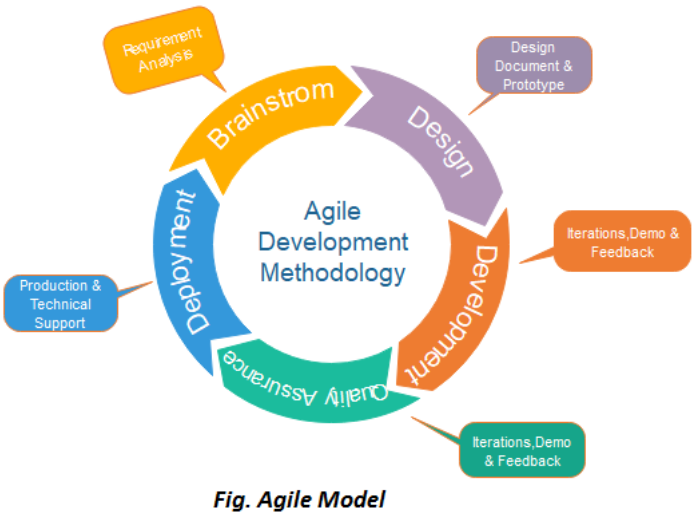

Figure 1 Agile Methodologies Features [11].

Since then, Agile methodologies have gained much popularity and success. The software industry had a huge shift from practicing traditional software development to now adopting Agile methodologies. There are several reasons why the software industry has chosen Agile over traditional models. Some of the reasons are faster product delivery, iterations, customer satisfaction, high product quality, etc.

These methods have led to major changes in how software is developed. Scrum [12] is now the most common framework for development in most countries, and other methods like Extreme Programming (XP) [7] and elements of lean software development and Kanban [13] are widely used also [10].

These agile processes, namely XP and Scrum are iterative, in which changes can be made according to the customer satisfaction provided through feedback. The characteristics on an agile process are [8]:

- **Iterative** – one of the main objectives of agile software processes is the satisfaction of customers, the principal requirements are multiple iterations.
- **Modularity -** is a key role in software development, where an agile process breaks down all the complete system into manageable pieces that are known as modules.
- **Time Boxing -** each module requires a time limit with the corresponding cycle because it is an agile process.
- **Parsimony** - in order to mitigate risks and achieve all the objectives it is important to use the minimum number of modules. In agile processes parsimony is required to mitigate risks and achieve the goals by a minimal number of modules.
- **Incremental -** an agile process requires to be developed in increments where each increment is independent of others. Lastly, all the increments create a complete system. As the agile process is iterative in nature, it requires the system to be developed in increments, each increment is independent of others, and at last all increments are integrated into a complete system.
- **Adaptive -** while the agile process is working there will be new risk. That is why there are adaptative characteristics that allow the process to adapt to new risks and

changes in real time. Due to the iterative nature of agile process new risks may occur. The adaptive characteristic of agile process allows adapting the processes to attack the new risks and allows real-time changes in the requirements.

- **Convergent -** All the risks associated with each increment are convergent in agile process by using an iterative and incremental approach.
- **Collaborative -** As agile process is modular in nature, it needs a good communication among the software development team. Different modules need to be integrated at the end of the software development process.

2.1. Continuous Integration

These new Agile methods brought a new concept of developing, testing, and deploying the software, to the industry.

Continuous Integration (CI) and Continuous Deployment (CD) were born with the Agile methods, and they have been changing the way to deliver, develop and test the software.

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software faster. [14]*"*

In the expression 'continuous integration', integration refers to the join software parts and continuous refers to the reduce as much as possible time-constraints. Several development methods cite certain activities as continuous integration. Continuous integration might be the supplement of each phase end, where 'continuous' refers to the way integration takes place during a phase of integration and tests; or it may be part of iterative methods or processes prescribing those modules and their interfaces are designed and documented prior to implementation.

2.2. Continuous Deployment

In order to deliver software to users as quickly as possible, Continuous Deployment (CD) and Continuous Integration (CI) give us some tools and methods to achieve this goal.

A deployment pipeline is, in essence, an automated implementation of the application's Build, Deploy, Test, and Release Process. Every organization will have its own implementation of the deployment pipeline (or several deployment pipelines according to specific projects), depending on their value stream for releasing software, but the principles that govern them do not vary.

In Figure 2, an example of a deployment pipeline is provided.

Figure 2 Continuous Integration Pipeline [14].

The pipeline is a result of several CI tools implemented. The deployment pipeline has three goals. First, it makes every part of the process of building, deploying, testing, and releasing software visible to everybody involved, aiding collaboration. Second, it improves feedback so that problems are identified, and thus resolved, as early in the process as possible. Lastly, it enables teams to deploy and release any version of their software to any environment at will through a fully automated process [15].

2.3. DevOps Concept

CI and CD have changed the developer's role lately - there are new roles that involve all the features and activities involved.

DevOps [16] is a new concept that came with the introduction of CI and CD practices. This role is specific for those that have transversal knowledge of Development, Quality Assurance [17] and Operations.

Thus, the concept of a DevOps tells that is a combination of Developers and Operations (Figure 3), where these kinds of developers, can do the developing, testing and infrastructure of operation activities.

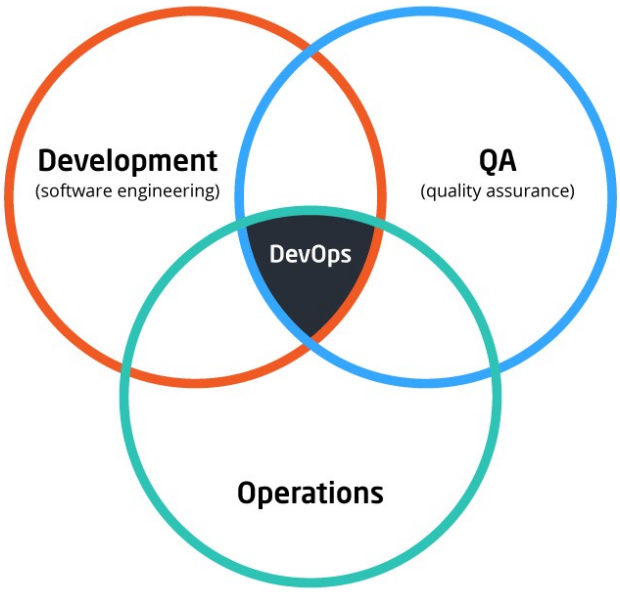

Figure 3 The adjustments DevOps has brought to software delivery [18].

There are some important concepts that let the DevOps concept emerge:

Infrastructure as Code - Server environments are detailed in manifest files, that are stored in repositories. These ones can be changed by any team member and reused multiple times to give the required infrastructure for testing or building the code[19].

Continuous Integration – It is important to have feedback, test and support in all kinds of different software and development phases, so continuous integration allows DevOps to have all this information that can help to improve the software.

Continuous Delivery – The way to present and deliver the software is important, that is why continuous delivery grants the DevOps a faster way to hand over the final product to the costumer

With the appearance of the DevOps concept, there were new roles that help to understand and handle the new features of the DevOps approach [19]. There are some roles such as: (Figure 4).

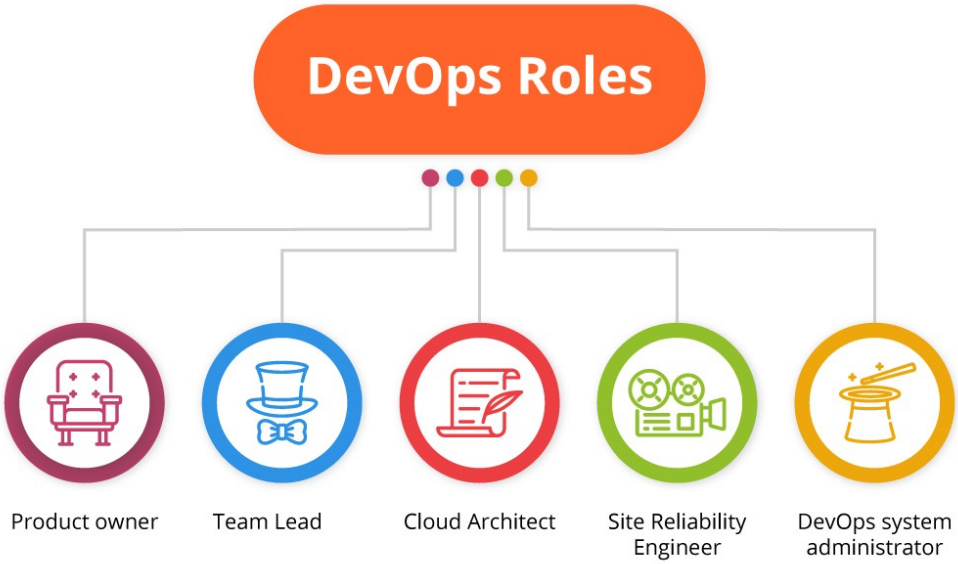

Figure 4 DevOps Roles [19].

Product owner - The software receives new requirements in any software development phase, so the cooperation between the costumer and the DevOps team is required. That interaction lets the development team to identify new requirements and look up new improvements for the app in production. The person who does the cooperation is the Product Owner.

Team Lead - This person is capable of having a general overview of the project with many responsibilities and it is in charge of delegating the DevOps tasks across the team.

Cloud Architect – Is related to the experience with building and cloud infrastructure. This person has to use all the hands-on experience to support different kind of software.

Site Reliability Engineer (SRE) – A software must be stable and interrupted availability across its life, this role has to guarantee to the customer the reliability of the software.

DevOps system administrator - One of the principal DevOps roles, as cloud monitoring represents more than a half of all DevOps tasks and time. Each team member has to be able to do the support tasks, these are the daily activities for the support administrator.

2.4. Continuous Testing

In order to ensure high quality and reliability, projects should deploy Continuous Testing [20], which involves running many different types of tests (automated and manual) throughout the software delivery process to repeatedly validate and improve the quality of the software that is being built.

The Figure 5 shows the different kinds of tests and the approach (manual vs test automation) to run [10]:

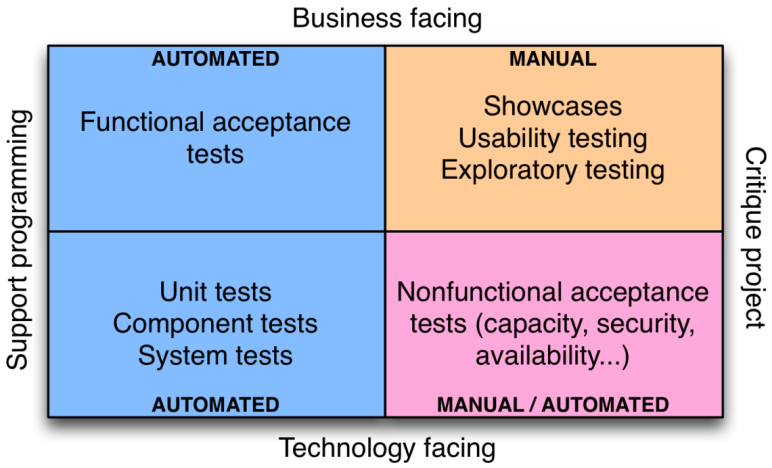

Figure 5 Manual vs Automated Test approaches [21].

As we can see in Figure 5, in the top-left, the recommended approach to functional acceptance tests is to automate most or all of them.

The goal of building software in Continuous Delivery mode is to keep the code ready for deployment to production at any time. Non-functional tests should be processed similar to functional tests and they should be executed to make sure the deployed code complies with not just functional issues but also performance, security, and other non-functional issues [21].

Non-functional testing (NFT) – These are the kind of tests used for test checking non-functional aspects (performance, usability, reliability, etc.) of a software application.

An example of non-functional test would be to check how many people can simultaneously login into a software [22]. The Figure 6 shows some kinds of NFT.

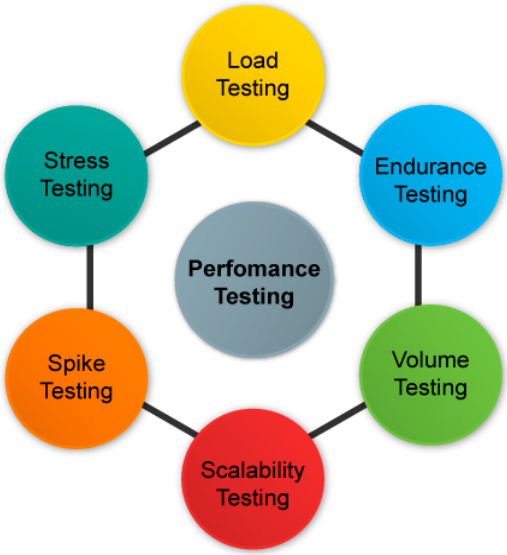

Figure 6 Non-Functional Test [23].

2.4.1. Non-Functional Tests in Continuous Testing mode

The two major categories of non-functional testing are Performance Testing [24] (includes Load, Stress etc.) and Security Testing (Figure 7). There are others such as Usability Testing (Figure 8), which is also considered non-functional.

One of the biggest challenges in designing non-functional tests is usually the elicitation of requirements for non-functional aspects of the software which are not as straight forward compared to the functional behavior of the software. This fact combined with other ones such as time and budget, usually prevents non-functional tests from being executed in continuous testing mode.

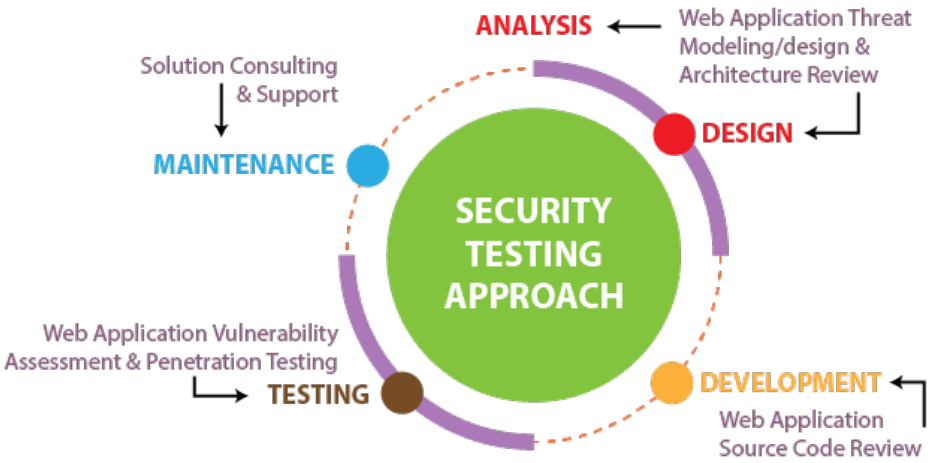

Figure 7 Security Testing Approach [25].

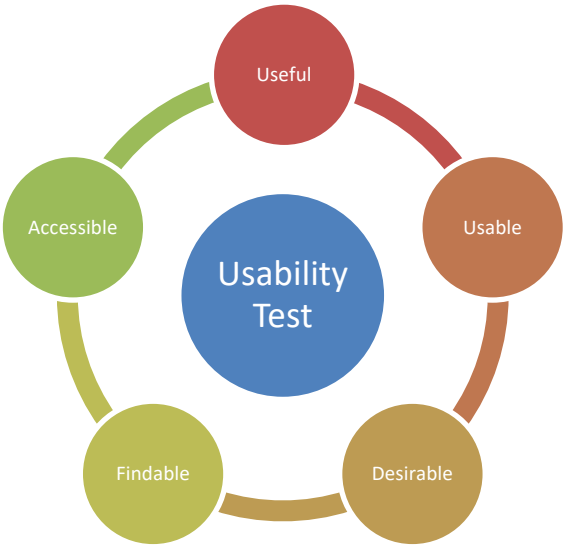

Figure 8 Usability Tests Approach [26].

2.4.2. Non-Functional Requirements Tests on the Pipeline

Non-functional requirements [22] are a difficult area because they make technical people provide more information into their analysis, these fact may distract them from the business value they are asked to deliver.

Technical people must work closely with customers and users to determine the most important parts of the application and define detailed non-functional requirements. Once this work has been done, the team can decide the correct architecture to use for the application and create requirements and acceptance criteria, the nonfunctional requirements will be gotten in the same way that functional requirements are captured and at the same time. After that, the team needs to create and maintain automated tests to ensure that these requirements are achieved. These tests should be run as part of deployment pipeline, every time a change to the application, infrastructure, or configuration is done.

At the beginning of the software lifecycle the team will take some time in initial requirements and architecture. One of the activities to be considered as part of requirements is to recognize non-functional requirement (NFRs), also called quality of service (QoS) or simply quality attributes. These NFRs should be captured by someone and implemented during all phases of any software development process. It is not enough to simply implement the NFRs, the team should validate that they have taken and identified them properly.

It is important to verify the NFRs in the pipeline. Therefore, there should be test regarding the NFRs.

The philosophy of agile validation means that all the member of a team should test the product. The normal approach is the team itself is responsible for validating its own work. The perfect approach will be to test the work of another person.

Developers should be performing regression testing to the best of their ability, adopting a continuous integration (CI) strategy in which the regression test suite(s) are run automatically many times a day [24].

There are some types of NFRs which require significant expertise to be addressed properly: NFRs pertaining to security, usability, and reliability, for example (see Figure 6). To validate these types of requirements, to identify them, requires skills and sometimes specialized tooling.

When this situation occurs, it is recommendable that the group of DevOps that have knowledge on these tools, focus on these kinds of testing.

Another strategy for validating NFRs regarding code analysis, dynamic and static, is the use of tools available that can address NFR types such as security, performance, and more. These tools will not only identify potential problems with your code, but many of them will also provide summaries of what they found, metrics that you can leverage in your automated project dashboards [24].

3. Research Design

The baseline for the company previously to this research was established. This means that it already had and practiced an agile methodology that involves some features of DevOps. Therefore, the goals of the project are the following:

- Conduct courses and capacitation to introduce the concepts of CI and CD (Figure 9).
- Analyze the metrics and measures of CI, this will be analyzed before the implementation.
- Select the metrics and measures to analyze with a project of Saviasoft.
- Analyze each metric selected and describe the results, after the implementation.
- Implement CI and CD on each phase of the software development.

This chapter will describe how the research was conducted. In order to implement the new features and methodology in the company, the following steps occurred:

3.1. Planification of the Trainings

The courses involved the explanation of a new topic or the use of new tools, for example there was given a training of CI, CD, and CDE. See Figure 9.

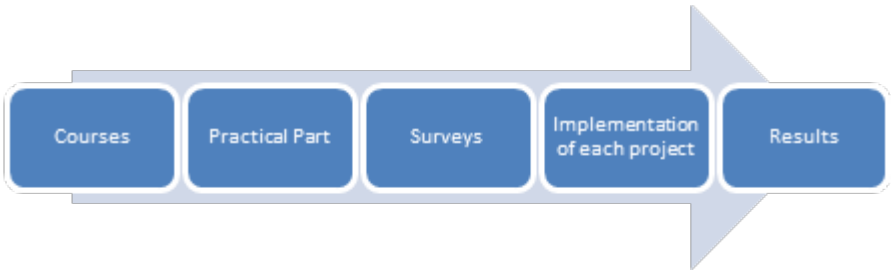

Figure 9 Courses workflow.

3.2. CI Implementation

This decertation will implement CI considering the following steps:

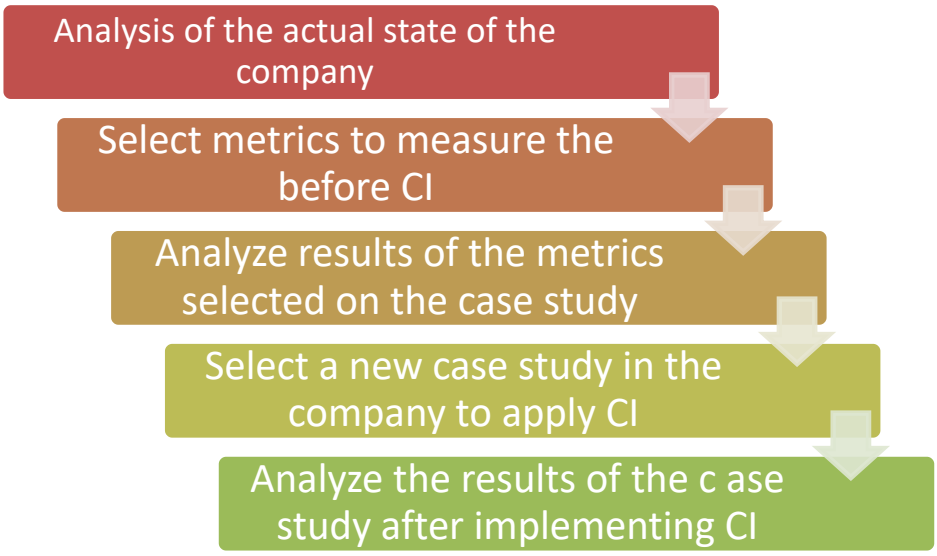

Figure 10 Steps of the project implementation.

3.3. Baseline regarding the development process in the company

The company used an agile methodology that has some features of Scrum.

The phases of the baseline methodology identified, are:

Requirements - Define the requirements for the iteration based on the product backlog, customer, and stakeholder feedback.

On this phase the developers and architects have a meeting after the definition with the client. Each developer selects the task that they want to do.

Development - Design and develop software based on the defined requirements. For the Quality Assurance the company uses a software that analyses the code (SonarQube [27]).

The developers must have a software that analyses the quality of the code, if it is possible, they should have installed a plug-in on their IDE to have feedback by the tool while they code.

Testing – Internal and External testing. This phase is where the customer and developers test the software and give feedback to the company.

The first stage of this phase is to do the alfa testing, on this phase the developers test their own code, most of the tests that are managed by the developers accomplish the Functional Requirements.

The second stage of this phase is beta testing, the client tests the final product, and these kinds of tests achieve a small part of the non-functional requirements, on this phase the company also acquire a feedback of the usability of the product and other nonfunctional requirements. Not all the non-functional requirements depend on the developers, the accessibility depends on the internal network of the client or the internet connection of the client.

Delivery – On this face the company delivers the final product (software, servers, dockers files, data bases, etc.). Once the client tests and accepts the software, the company proceeds to install all the infrastructure that the software needs. This phase depends a little bit on the clients because most of the time they are the ones that give to the company the available infrastructure that they have, e.g., most of the clients prefer to have their servers physical on their companies. Not every client prefers a cloud base approach.

Feedback - Accept customer and stakeholder feedback and work it into the requirements for the next iteration.

All software and tools used on each software phase are standards use by the company, unless the customer has their own standards, in this case the company uses the software standards of the customer. The company has been implementing this methodology with all its clients.

3.4. Metrics - Measure, Impact of Continuous Integration

In order to implement a new Continuous Integration methodology there were some steps that were done in the company. Also, in order to measure and document the impact of CI and CD in a software development company there are some metrics that help to track all phases of software development, such as:

Lead time per work item to production

The time that each item or subitem takes to be delivered to the client, in CI and CD involves also when the item is deployed in the production environment.

Cycle time to "done"

This measure appears when an item takes much more time that the estimated one, or when an item changes a lot in every iteration. It starts when there is all information to do an item and when the developers deliver, and the client accepts the item or items.

Escaped defects

This measure is when the item or items were done, and a bug is discovered after the delivery. Sometimes it is not a bug, it is unfinished work. This one is measured in number bugs found.

Total regression test time

It measures tracks the total time that takes to do a full regression test. It includes both manual and automated tests. Teams that have primarily manual tests will measure this in weeks or months. Teams that have primarily automated tests will measure this in minutes or hours.

Number of branches in version control

The number of branches also helps to track and measure, each branch involves a significant or a big change in the requirements, measuring and considering the number of branches helps to realize how long it will take to end the project or how risky is.

These metrics were chosen because these are the ones that fit on the agile methodology that the company has, also they give the perfect feedback to evaluate the change of developing.

3.5. Example of actual values for the metrics in projects of Saviasoft – to establish the baseline (before CI)

Saviasoft is a small company, and currently it has 7 developers. There is no operation or infrastructure area in the company, therefore all developers have been learning about the operational and infrastructure area. All company developers are DevOps because they have experience and knowledge in developing, testing and infrastructure area.

Saviasoft has 7 DevOps, 2 of them are software architects with more than 10 years of experience, three of them are senior developers with more than 5 years of experience, and there are 2 junior DevOps with more than 1 year of experience.

The pipeline that will be implemented is an extended process of the current methodology with the CI and CD issues.

4. Training

The results of the questionnaire on section (6.5) let us know the DevOps knowledge about CI and. CD. The results led to the planning of the following training sections:

The trainings will be made with the main tools that the company will use, the tools that will be explained in each training are going to be implemented in order to stablish a CI standard for the company. On each training will be explained the estimated time and how many people assisted.

- Introduction to CI.
- My first pipeline on Jenkins.
- Automated Test.
- JMeter.
- SonarQube.

4.1.1. Introduction to CI

This was a capacitation to all collaborators and CEOs of the company. This one has some introductory concepts about CI and CD, the objective of this capacitation was to give information about these new concepts.

On this training the following topics were explained:

- History - The history how the CI and CD were born.
- Continuous Delivery - General concepts of the Continuous Delivery, what does it involves and why was it born?
- CI - What is CI, principal concepts.
- Principal Issues of CI and CD - Principal characteristics, practices that involves CI and CD.
- How to apply CI and CD - Steps, practices, issues that a CI methodology needs in order to be implemented.

- CD - Principals concepts of CD.
- CI vs CD - Differences and Similitudes between CI and CD.
- Benefits - Principal benefits that come out after implementing a CI methodology.
- Principal tools of CI and CD - Principal tools that let the DevOps to implement CI and CD.

The slides of this training could be found on: Appendix C - Course: Introduction to CI & CD.

People: On this training all the DevOps of the company participated.

Time: This training took 45 minutes.

4.1.2. My first pipeline on Jenkins

This capacitation was about Jenkins and its fundamentals. In this one there was explained the use of Jenkins and its features.

In this capacitation it was explained:

- Jenkins Introduction – The DevOps learned the fundamental of Jenkins and how Jenkins works.
- How to create a new Jenkins Project – The DevOps learned how to create a new project on Jenkins.
- Jenkins File – There was explained how to create a Jenkins file (Figure 11). The DevOps received a Jenkins file example in order to know the elements that will be used. On the file there were stages, steps, post activities, etc.
- Git Credentials on Jenkins - There were explained how to add the git credentials in order to connect a git repository to the Jenkins project.
- Webhooks – The DevOps learned how to integrate the Webhooks with the git repository previously configured.

  How to execute the pipeline - There were explained how to execute and watch the log of the pipeline.

People: On this training there were all the DevOps of the company.

Time: This training took 45 minutes.

```
pipeline {
    agent any
    tools {
        maven 'maven-local'
    }
    stages {
        stage('Construir') {
            steps {
                sh 'mvn -f ilc-sce/pom.xml clean package'
            }
        }
        stage('Controlar calidad') {
            environment {
            scannerHome = tool 'saviasoft-sonar-scanner'
        }
            steps {
                withSonarQubeEnv('saviasoft-sonar') {
                sh "${scannerHome}/bin/sonar-scanner -Dproject.settings=ilc-sce/sonar-
project.properties"
            }

            timeout(time: 5, unit: 'MINUTES') {
              waitForQualityGate abortPipeline: true
            }
          }
        }
    stage('Respaldar componentes') {
            steps {
                sh 'mvn -f ilc-sce/pom.xml clean deploy -Dwildfly.deploy.skip=true'
            }
        }
    }
    post {
      success {
            mail bcc: '', body: "<b>Saviasoft CI</b><br>Project: ${env.JOB_NAME} <br>Build
Number: ${env.BUILD_NUMBER} <br> URL de build: ${env.BUILD_URL}", cc: '', charset: 'UTF-
8', from: '', mimeType: 'text/html', replyTo: '', subject: "Saviasoft CI - EXITO: Project name ->
${env.JOB_NAME}", to: "juan.ochoa@saviasoft.com";
        }
    }
}
```

Figure 11 Jenkins File example.

4.1.3. Automated Test

There was a capacitation about how to do automated test using the following tools:

- Cucumber.
- Selenium.
- IntelliJ Idea.
- JUnit.
- Katalon Recorder.

In this capacitation there was explained how to use Katalon recorder.

Katalon Recorder – Is an automation recorder that helps export Selenium WebDriver code. You can also record actions, capture web elements on web applications, play automated test cases, and do reporting quickly and easily [28].

In the capacitation Katalon was used to get the code in order to test it on Java. For the purpose of learning how to record all web activities (Figure 12), the DevOps test a software of the company that is on the cloud: www.a-vender.com.

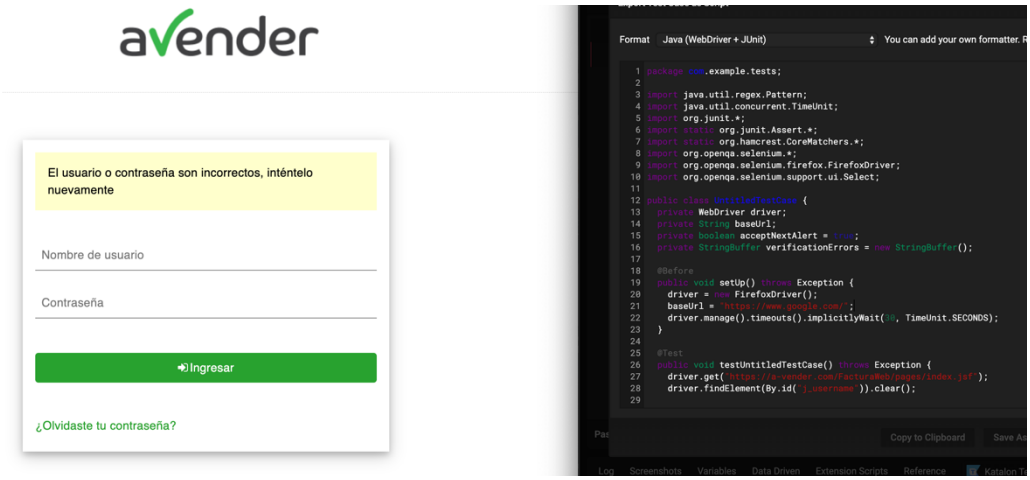

Figure 12 Use of Katalon Recorder.

With Katalon recorder the code was exported to Java, a list of prices was tested, the user clicks on the button edit and changes the price, after that the test verifies that the price was changed successfully. Example of the code for testing (Figure 13).

```
@Then("^Selecciono y edito el precio que deseo\\.$")
public void seleccionoYEditoElPrecioQueDeseo() {

    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-space(.)='CAJA GENERAL'])[2]/following::i[2]")
    driver.findElement(By.name("j_idt98:j_idt113:4:j_idt123")).click();
    driver.findElement(By.name("j_idt98:j_idt113:4:j_idt123")).clear();
    Integer valor = (int) (Math.random() * 50 + 1);
    driver.findElement(By.name("j_idt98:j_idt113:4:j_idt123")).sendKeys(valor.toString());
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-space(.)='CAJA GENERAL'])[2]/following::i[4]")
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    System.out.println("valor: ********** "+ driver.findElement(By.xpath("//*[@id=\"j_idt98:j_idt113:4:valor\"]")).getTe
    Assert.assertEquals(valor.toString(), driver.findElement(By.xpath("//*[@id=\"j_idt98:j_idt113:4:valor\"]")).getText());

}
```

Figure 13 Code of the Automated Tests.

The automated tests are in a git repository so they could be part of a pipeline. A new item of the pipeline was added.

People: On this training there were all the DevOps of the company.

Time: This training took 1 hour and 45 minutes.

4.1.4. JMeter

This training explained how to do a proxy recorder of a company software, these trainings aimed to test nonfunctional requirements.

The main steps to accomplish this course were:

- Create a JMeter file that record the steps to test NFR.
- Create an additional step to run the JMeter file inside the Jenkins server.
- Get the results on JMeter.

In order to create the JMeter file, it is necessary to open the jmeter.sh that is on the folder of apache-jmeter. It was explained how to record all activities by proxy configuration (Figure 14).

There we configured the login and 2 main screens of the selected software which should be saved to a file.

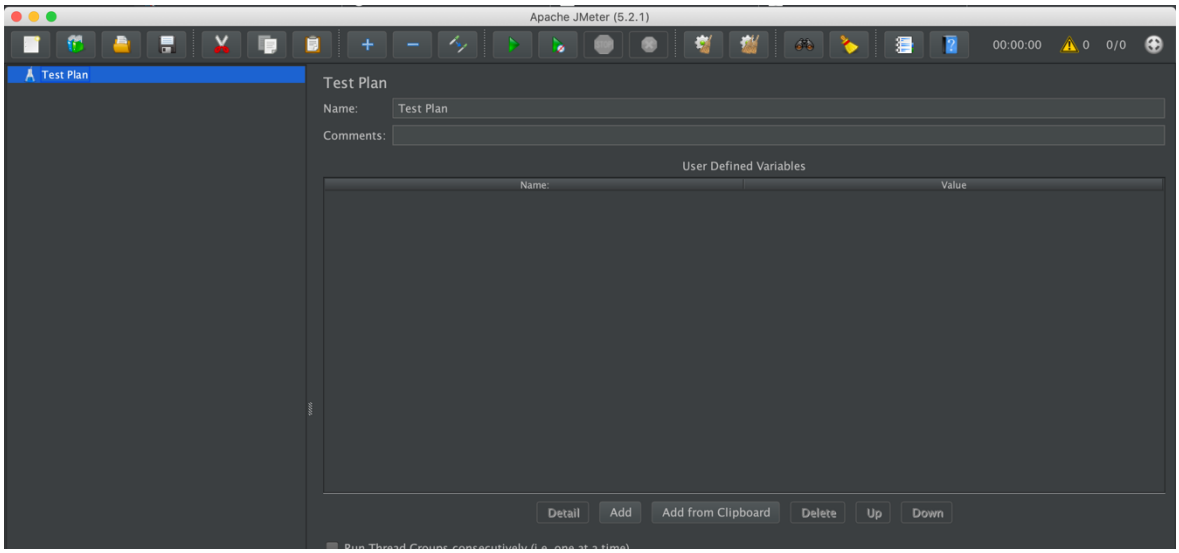

Figure 14 Jmeter Software.

On Jenkins we create a new item that will run the JMeter test, and we will have the results on the pipeline. There were added a shell in order to execute the JMeter file.

There are 2 options to get the statistics: make the configure on the Jenkins server or configure on the server that is running the software. On this course there was configured on the Jenkins server (Figure 15).

Figure 15 Jmeter commands.

Once we execute the step, the result is showed on a statistic graphic as the Figure 16 shows.

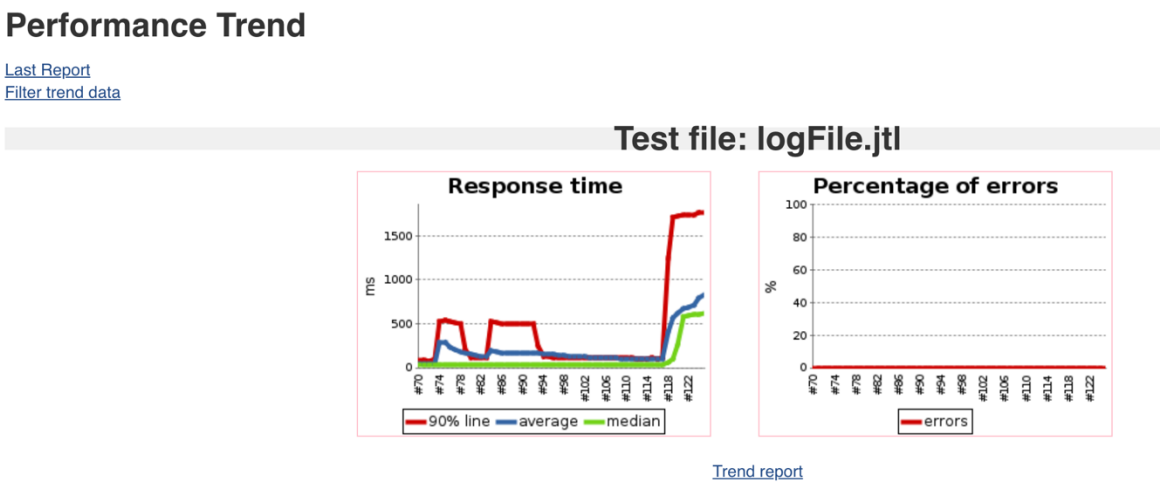

Figure 16 Jmeter results: Statistics.

People: On this training there were all the DevOps of the company.

Time: This training took 60 minutes.

4.1.5. SonarQube

This course was about a new item on the Jenkins pipeline that runs a SonarQube [27] test to analyze the code. SonarQube is already used by the company but not on a Jenkins pipeline.

SonarQube - Is an open-source platform developed for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on more than 20 programming languages. SonarQube offers reports on duplicated code, coding standard, code coverage, code complexity, comments, bugs, and security vulnerabilities [3].

The same file that was used to create the first pipeline was used in this course; another step to create the sonar step was added to the pipeline.

```
        steps {
          withSonarQubeEnv('saviasoft-sonar') {
          sh "${scannerHome}/bin/sonar-scanner -Dproject.settings=ilc-
sce/sonar-project.properties"
            }

            timeout(time: 5, unit: 'MINUTES') {
          waitForQualityGate abortPipeline: true
            }
        }
```

Figure 17 Step of SonarQube on the pipeline.

This step will analyze all the code that is on the software. It is necessary to tell Jenkins which code will be analyzed. On the pipeline it should be described which git repository will be analyzed. There were configured credentials to connect to the git repository.

In order to tell the Jenkins which classes should be teste it is necessary to add a properties file on the principal project of the Maven project.

```
#Required metadata
sonar.projectKey=ilc-sce
sonar.projectName=ilc-sce
sonar.projectVersion=4.10-SNAPSHOT
sonar.language=java

#Multiple modules
sonar.modules=ejb, web

#ejb
ejb.sonar.projectName=Ilc ejb
ejb.sonar.projectBaseDir=.
ejb.sonar.sources=./ilc-sce/ilc-sce-ejb/src/main/java
ejb.sonar.java.binaries=./ilc-sce/ilc-sce-ejb/target/classes

#web
web.sonar.projectName=Ilc web
web.sonar.projectBaseDir=.
web.sonar.sources=./ilc-sce/ilc-sce-web/src/main/java
web.sonar.java.binaries=./ilc-sce/ilc-sce-web/target/classes
#web.sonar.exclusions=**/*.js, **/*.xhtml, **/*.css, **/*.jar
```

Figure 18 Properties on the principal project.

The pipeline was executed, and we could see the execution (see Figure 19).

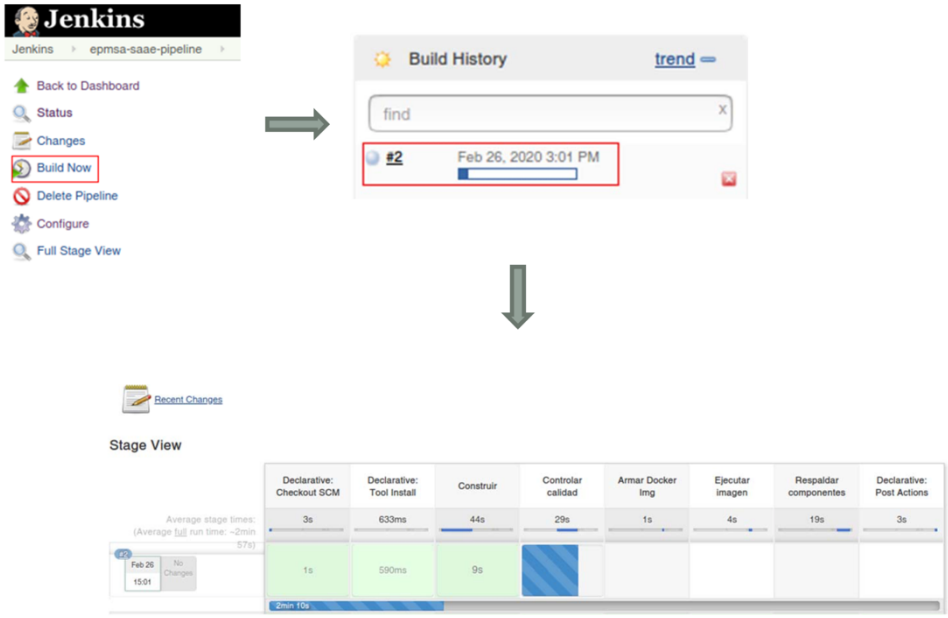

Figure 19 Execution of the pipeline with SonarQube.

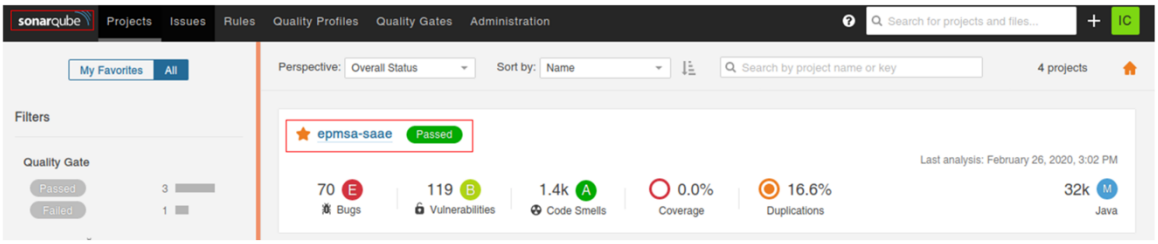

Figure 20 Result of the SonarQube execution.

People: On this training there were all the DevOps of the company.

Time: This training took 45 minutes.

5. CI Implementation

In order to compare [29] the results a new case of study was selected. The case study selected was: Formados.

Formados is a software that seeks to strengthen dual training as a shared professional training methodology (theoretical - practical) which provides a holistic education. Within this, Higher Education Institutions (HEIs) develop and teach the theoretical bases of the profession and training companies they develop and form the practical bases of the profession for students. All of this within a real work environment, which enables qualified human talent to be trained with the required professional skills, thereby improving the productivity and competitiveness of the country's productive sector.

The main goal of the software is: develop a Platform to achieve a more efficient management of the Dual Training management processes and the proper documentation facilitating the work of business tutors, academic tutors and dual training coordinators of the chambers, which manages to reduce the work times of each one of these users and that allows the fulfillment of their responsibilities in the established terms as well as the generation of status and results reports.

The case study has the following modules:

- Institutions: This module let the user register.
    - Enterprises.
    - Colleges or Universities.
    - Resources.
    - Agreements.

- Security: Each role has his own functionality. There are the following roles:
    - Formados Role.
    - Student.
    - Enterprise Role.
    - Academic Tutor.
    - Training Tutor.
    - Chamber Coordinator

- Academic Coordinator.
- Dual Training Company Manager.

- Operations:
  - Enterprises
    a. Tutor assignation to a specific area of the company
    b. Planification of the learning and the project that the student will be assigned
    c. Tracing of the dual formation process
    d. Students' evaluation
  - Colleges/Universities
    a. Creation and managing of each career
    b. Academic coordinators assignation and tutor for each career
    c. Planification of the syllabus for each career
    d. Students by career
    e. Students' assignation to the academic tutor
    f. Grades registration
    g. Tracing of the dual formation process
    h. Review of the practical learning and enterprise project
- Manage:
  - Parameters for the application
  - Workflows and Task of the process
  - Documents templates
  - Parameters for the enterprises
  - Parameters for each career for the enterprises
- Students:
  - Enterprises Role
    a. Students' selection
    b. CV of the student
    c. Results of Psychometric Test
  - Formados Role
    a. Selection of the enterprises
    b. Enterprise's change
    c. Student History
    d. Results of Psychometric Test

- Student Role
    a. Student's account.
    b. CV
    c. Psychometric Test
    d. Register of the activities in the enterprises
    e. Survey
    f. Grades

For the purpose of implement CI on this project there were done the following steps:

1. Create a new Jenkins project and create a Jenkins file on it.
2. Configure SonarQube to test the code of this software.
3. Create and develop the automated test.
4. Develop automated test.
5. Test the NFR with JMeter.

Formados software was selected for the case of study because it was a new software implementation. Therefore, it was perfect for the implementation of the case of study, it was easy to analyze the same features for both projects.

5.1. Jenkins Pipeline

In order to implement CI on the case of study selected for the analysis there was created a Jenkins file to describe the steps of the project.

The Jenkins [30] file (Figure 21) will be on the principal git repository of the case study.

```
pipeline {
    agent any
    tools {
        maven 'maven-local'
    }
    stages {

        stage('Construir') {
           steps {
               sh 'mvn -U -f util/pom.xml clean package'
               sh 'mvn -U -f seguridad/pom.xml clean package'
               sh 'mvn -U -f connect/pom.xml clean package'
               sh 'mvn -U -f connect-interfaz/pom.xml clean package'
           }
        }

        stage('Controlar calidad') {
           environment {
                   scannerHome = tool 'saviasoft-sonar-scanner'
           }
           steps {
              withSonarQubeEnv('saviasoft-sonar') {
                sh "${scannerHome}/bin/sonar-scanner -Dproject.settings=connect/sonar-project.properties"
              }

              timeout(time: 5, unit: 'MINUTES') {
                waitForQualityGate abortPipeline: true
              }
           }
        }
        stage('Crea Imagen docker'){
            steps {
            echo "imagen de docker"
            }
        }

        stage('Test Automatizados'){
            steps {
                echo " test automaticos"
                sh 'mvn -f pruebas.automaticas/pom.xml clean package'
                junit 'pruebas.automaticas/target/surefire-reports/*.xm
```

Figure 21 Formados Jenkins File.

The Pipeline [30] of the project (Figure 22) has the following steps:

1. Build the Maven project.
2. Analyze code quality with SonarQube.
3. Create a docker image.
4. Run automated test.
5. Run Non-Functional Test.
6. Send results.

Figure 22 Pipeline Formados Project.

5.2. SonarQube Analysis

One of the aims of this dissertation was to improve the software quality of the company. Therefore, SonarQube [3] was used to analyze the code. SonarQube was installed on a cloud server, the configuration file is on the repository of the project.

The study case was developed with a micro services standard; hence the case of study has 4 principals projects, the ejb project and the web projects that contains the web services. On the SonarQube configuration file (Figure 23) there are described each project that is analyzed.

```
#Required metadata
sonar.projectKey=formados
sonar.projectName=formados
sonar.projectVersion=0.0.1-SNAPSHOT
sonar.language=java

#Multiple modules
sonar.modules=seguridad-ejb, seguridad-web, connect-ejb, connect-web

#seguridad-ejb
seguridad-ejb.sonar.projectName=seguridad-ejb
seguridad-ejb.sonar.projectBaseDir=.
seguridad-ejb.sonar.sources=./seguridad/seguridad-ejb/src/main/java
seguridad-ejb.sonar.java.binaries=./seguridad/seguridad-ejb/target/classes

#seguridad-web
seguridad-web.sonar.projectName=seguridad-web
seguridad-web.sonar.projectBaseDir=.
seguridad-web.sonar.sources=./seguridad/seguridad-web/src/main/java
seguridad-web.sonar.java.binaries=./seguridad/seguridad-web/target/classes

#connect-ejb
connect-ejb.sonar.projectName=connect-ejb
connect-ejb.sonar.projectBaseDir=.
connect-ejb.sonar.sources=./connect/connect-ejb/src/main/java
connect-ejb.sonar.java.binaries=./connect/connect-ejb/target/classes

#connect-web
connect-web.sonar.projectName=connect-web
connect-web.sonar.projectBaseDir=.
connect-web.sonar.sources=./connect/connect-web/src/main/java
connect-web.sonar.java.binaries=./connect/connect-web/target/classes

```

Figure 23 Formados SonarQube File Properties.

The result of the SonarQube analysis (Figure 24) is good, since the case of study has 0 bugs and 0 vulnerabilities.

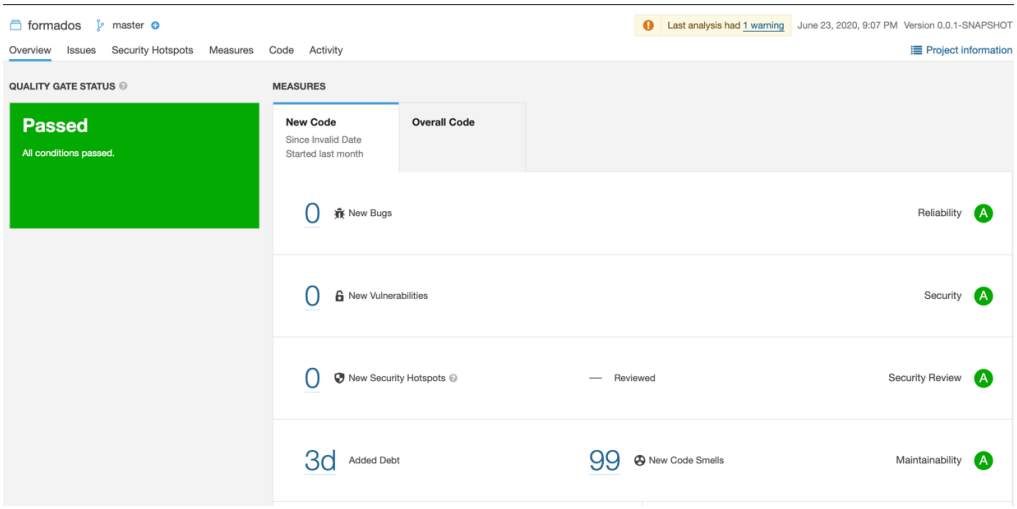

Figure 24 SonarQube Result of the study case.

5.3. Docker Implementation

The case of study has 4 principal modules:

- Seguridad
- Mail
- Connect
- Connect-interfaz

The case of study selected has 4 dockers for its implementation, in this dissertation we will cite 2 of them.

Each docker [31] will have a Wildfly 18 with the determinate ejb deployed. On the (Figure 25) there is the detail of the DockerFile of the Seguridad module.

```
FROM jboss/wildfly:18.0.1.Final
RUN /opt/jboss/wildfly/bin/add-user.sh admin admin --silent
CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0", "-bmanagement", "0.0.0.0"]

USER root

COPY modules/saviasoft /opt/jboss/wildfly/modules/system/layers/base/com/saviasoft
RUN chown -R jboss:jboss /opt/jboss/wildfly/modules/system/layers/base/com/saviasoft

COPY modules/microsoft /opt/jboss/wildfly/modules/system/layers/base/microsoft
RUN chown -R jboss:jboss /opt/jboss/wildfly/modules/system/layers/base/microsoft

COPY standalone.xml /opt/jboss/wildfly/standalone/configuration
RUN chown jboss:jboss /opt/jboss/wildfly/standalone/configuration/standalone.xml

COPY postgresql-42.2.12.jar /opt/jboss/wildfly/standalone/deployments
RUN chown jboss:jboss /opt/jboss/wildfly/standalone/deployments/postgresql-42.2.12.jar

COPY saviasoft-mail-ejb-1.2.jar /opt/jboss/wildfly/standalone/deployments
RUN chown jboss:jboss /opt/jboss/wildfly/standalone/deployments/saviasoft-mail-ejb-1.2.jar
```

Figure 25 DockerFile of the mail module.

5.4. Automated Test Implementation

A new stage on the pipeline was created in order to implement the automated test. This stage was made using cucumber [32] standard to define the features and the scenarios.

On the (Figure 26) there is an example of a feature done for the case of study.

```
Feature: Test de Hoja de Vida del estudiante

  Scenario: Test Agregar Experiencia Laboral
    Given Click  en hoja de Vida para abrir el Pop up
    When Click en el tab de expeiencia laboral
    Then El nombre en la tab debe ser "Experiencia Laboral"
    Given Se proporcionan datos de una nueva experiencia laboral
    Then Se verifican los nuevos datos ingresados

    Given Se va a editar el registro creado
    When Se cambiala informacion del registro seleccionado
    Then Se verifica que se a editado correctamente el registro

    Given Se va a eliminar el registro creado
    When Se confirma que se va a eliminar
    Then Se verifica que se elimino el registro
```

Figure 26 Feature file of the hoja de vida steps.

As the case of study is done in java, the automated tests also were done in java, selenium web driver was used to test the case of study, chromedriver [33] was used to navigate through the web pages of the study case. The (Figure 27) shows the source code of the java code.

```
@When("Se cambiala informacion del registro seleccionado")
public void se_cambiala_informacion_del_registro_seleccionado() {
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    wait.until(ExpectedConditions.elementToBeClickable(By.id("form-experiencia-laboral:txt-inst")));
    wait.until(ExpectedConditions.elementToBeClickable(By.id("form-experiencia-laboral:txt-anio-desde")));
    driver.findElement(By.id("form-experiencia-laboral:txt-inst")).clear();
    driver.findElement(By.id("form-experiencia-laboral:txt-inst")).sendKeys(institucionCambio);
    driver.findElement(By.id("form-experiencia-laboral:txt-funciones")).sendKeys("Funciones en empresatest");
    driver.findElement(By.id("form-experiencia-laboral:txt-anio-desde")).sendKeys("2012");
    driver.findElement(By.id("form-experiencia-laboral:txt-anio-hasta")).sendKeys("2014");
    driver.findElement(By.id("form-experiencia-laboral:btn-guardar-exp")).click();
}

@Then("Se verifica que se a editado correctamente el registro")
public void se_verifica_que_se_a_editado_correctamente_el_registro() {
    cargarDeNuevoPagina();
    WebElement tabla = driver.findElement(By.id("form-experiencia-laboral:experienciasLista"));
    List<WebElement> datosT = tabla.findElements(By.className("col-datos"));
    numeroFilas = datosT.size() - 1;
    boolean verificaCambioRegistro = false;
    for (WebElement we : datosT) {
        String[] columnas = we.getText().split("\n");
        for (String s : columnas) {
            if (institucionCambio.equals(s)) {
                verificaCambioRegistro = true;
                break;
            }
        }

    }
    assertTrue(verificaCambioRegistro);
}
```

Figure 27 Automated test code.

5.5. Non-Functional Test

In order to implement non-functional test JMeter [34] was used to implement them.

On the Figure 28 there is the log in test of the case of study.

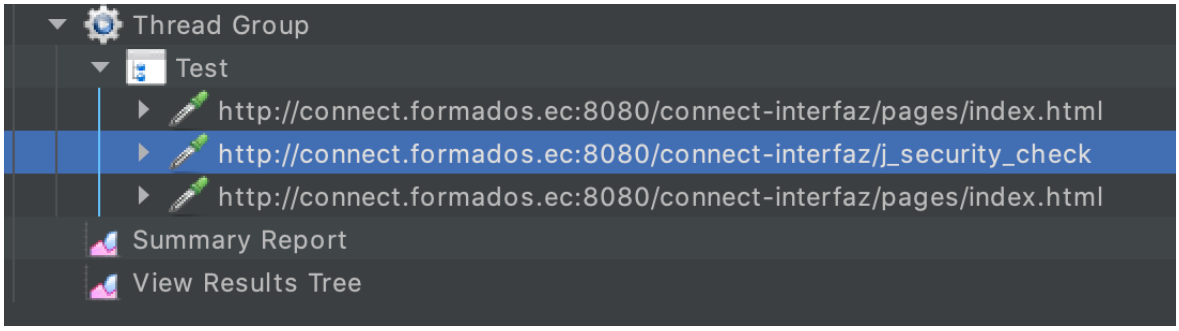

Figure 28 JMeter test of the log in.

The results of non-functional tests [35] shows how reliable the software is, in the Figure 29 there are the results of the test done to the case of study. these results show the performance of the software, it gave the response time, latency, and so on.

Sample #	Start Time	Thread Name	Label	Sample Time(...	Status	Bytes	Sent Bytes	Latency	Connect Tim...
1	19:18:59.946	Thread Grou...	http://connec...	945		252821	5073	311	204
2	19:19:00.051	Thread Grou...	http://connec...	902		252821	5073	212	107
3	19:19:00.150	Thread Grou...	http://connec...	838		252817	5073	202	95
4	19:19:00.251	Thread Grou...	http://connec...	965		252819	5073	198	98
5	19:19:00.347	Thread Grou...	http://connec...	1068		252815	5073	203	98
6	19:19:00.446	Thread Grou...	http://connec...	1060		252819	5073	209	105
7	19:19:00.546	Thread Grou...	http://connec...	1089		252821	5073	184	90
8	19:19:00.647	Thread Grou...	http://connec...	1086		252819	5073	186	91
9	19:19:00.747	Thread Grou...	http://connec...	1042		252823	5073	186	88
10	19:19:00.846	Thread Grou...	http://connec...	1089		252823	5073	292	92
11	19:19:07.682	Thread Grou...	http://connec...	1688		638841	13789	131	0
12	19:19:09.443	Thread Grou...	http://connec...	841		40814	14969	197	0
13	19:19:00.546	Thread Grou...	Test	3618		932476	33831	512	90
14	19:19:08.538	Thread Grou...	http://connec...	1755		638841	13789	126	0
15	19:19:10.380	Thread Grou...	http://connec...	919		40814	14969	177	0
16	19:19:00.647	Thread Grou...	Test	3760		932474	33831	489	91
17	19:19:10.531	Thread Grou...	http://connec...	1737		638841	13789	116	0
18	19:19:12.281	Thread Grou...	http://connec...	936		40811	14969	208	0
19	19:18:59.946	Thread Grou...	Test	3618		932473	33831	635	204

Figure 29 Results of the Jmeter test.

6. Results

The results [36] presented on this chapter are according to the metrics selected before; these ones will give to the company valuable information. This information will be compared with the results before CI.

For the purpose of comparison, a module of a whole project will be analyzed as a base line, there will be described all the methodology and metrics that Saviasoft uses.

6.1. Case of Study Analyzed

The case of study selected to analyze was a bill system, the module analyzed was: Liquidaciones, the functional requirements of the selected module will be explained following. Liquidacion de Compras - Is the document that is given to a person when the company must return some expenses that the company must pay but the person paid by himself.

Next, we present some examples of the requirements addressed in this feature.

Requirements:

- The user will be able to do a “Liquidacion de Compras” in the software.
- The user will be able to select the supplier with an autocomplete component.
- The user will be able to insert a bill as an element, each bill has the same data of the bills in the system for example: product, price, discount, customer, etc.
- The user will be able to fetch all the “Liquidacion de Compras” on a table.
- The user will be able to download the xml and the pdf format of the “Liquidacion”.
- The user will be able to filter the “Liquidacion de Compras” of the table, depending on the customer id, date of issue, number, state of “Liquidacion de Compras”.
- The user will be able to import a bill as a detail of the “Liquidacion de Compras”.
- Send the data of a “Liquidacion de Compras” to validate, using a web service.

Web services to receive data from the client

- A new web service that receives the data of a "Liquidacion de Compras" and this process will save on the database all data.
- A service that returns the state of a single or a group of "Liquidacion de Compras".

Features for the client that uses the web services

- Access to a web page that show all the "Liquidacion de Compras" that the client has. The page will have most of the functions that the internal system has.

6.2. Results Before CI

In order to estimate the time for a project, the company uses a Gantt diagram [37] to describe all the activities, like this the leaders of each project figure out the time that the project will take. In order to analyze the results of the actual methodology a module of a whole system was analyzed.

A work item is a task to do for a developer, usually each task is a whole requirement.

Work Items / Tas	Estimation	Real Time
A-vender-Software	**18 days**	**17 days**
Tables, persistence, java services for the new tables	2 days	1 days
Services for the new features (New module)	8 days	6 days
Web Services to receive data from the client.	8 days	10 days
Connection from Actuaria	**4 days**	**4 days**
Web Services to send the data to the A vender software	4 days	4 days
Reports	**2 days**	**2 days**
PDF of the new document	2 days	2 days
Tests	**3 days**	**8 days**
Document of each test case	0.5 days	0.5 days
Revision and acceptance of the test case document	0.5 days	0.5 days
Functional test	1 day	4 days
Improvements and corrections	1 day	3 days
Release to production	**3 days**	**5 days**
Changes to the actual library (jar)	1 day	1 day
Release to production	2 days	2 days
Tracing after release the module	1 day	2 days

Table 1 Requirement's estimation base line project.

Each metric will show the time that took to do it.

Lead time per work item to production (23 days)

Each item has his own estimation and the real time, we could see that the estimation and real do not change a lot.

The company sometimes delivers each item isolated, but usually each item must be delivered as a module and a module has more than one work item.

Defect fix times (8 days)

The bugs are solved as soon as they are found. There are items just for testing in the estimation.

Fix Times	
A-vender-Software	**Number of Bugs**
Tables, persistence, java services for the new tables	0
Services for the new features (New module)	6
Web Services to receive data from the client	10
Connection from Actuaria	
Web Services to send the data to the A vender software	4
Reports	
PDF of the new document	2

Table 2 Errors and bugs of the base line project.

Total regression test time (2 days)

There is not regression or automated test.

The Table 2 Errors and bugs of the base line project. shows the test that this case of study has.

Kind of test	Number
Unit Test	5
Procedure Test	3
Total	8

Table 3 Test before CI.

Test Method: Manual.

Number of branches in version control

Number of branches: 5. As we have 5 branches there were 5 different modules or different process to release to production.

This project has the number of branches depending on each module. This module was developed on a new branch.

Release to production (1 day)

The software is released to production manually.

In order to release this module to production the company does the following steps:

- Alpha and Beta testing.
- Assure all requirements.
- Generate the ear with the new features.
- Update the database with all new tables and columns.
- Stop the wildfly server on the cloud and replace the ear file that is inside the deployments folder.
- Start the wildfly server with the new ear file.

This time to production is the first time that the software was released to production. In the future, each time that the software needs an update the developers team takes 1 hour.

6.3. Results After CI

In order to get accurate results, another case of study project in the company was analyzed. We selected the same metrics that were analyzed before CI.

Depending on the developers and their projects they have implemented each tool and feature of CI. Every project of the company has a pipeline on Jenkins to connect the git repository now. All new projects on the company have the CI tools, former projects that are already in production they will have the CI tools step by step depending on the complexity of them.

The pipeline of each project has the following steps:

- Build the code with maven.
- Analyze the code with SonarQube.
- Build the docker image.
- Execute docker image.
- Run automated test.
- Run Non-functional test (JMeter).
- Deploy to production.
- Send results of the build.

Lead time per work item to production (13 days)

As can be seen in Table 4 the team estimated 6 days for the work item related to planification of the learning and the project that the student will be assigned to, but the work item took only 5 days “Real Time”. The other works items were very well estimated, since the estimation time was equal to the real time.

Work Items	**Estimation**	**Real Time**
Formados Software - Operations	**14 days**	**13 days**
Tutor assignation to a specific area of the company	2 days	2 days
Planification of the learning and the project that the student will be assigned	6 days	5 days
Tracing of the dual formation process	4 days	4 days
Students' evaluation	2 days	2 days

Table 4 Requirement's estimation – Operations.

Work Items	**Estimation**	**Real Time**
Formados Software – Curriculum	**10 days**	**10 days**
General information: address, telephone, so on.	3 days	3 days
Academic Formation	2 days	2 days
Work Experience	2 days	2 days
Languages	1 days	1 days
Personal and work experience	1 days	1 days
Aptitudes	1 days	1 days

Table 5 Requirement's estimation – Curriculum.

Work Items	Estimation	Real Time
Formados Software – Training Companies	**12 days**	**12 days**
General Information	2 days	2 days
Contact Information	1 days	1 days
Legal representative	1 days	1 days
Offices - Addresses	2 days	2 days
Tutors	2 days	2 days
Areas	1 days	1 days
Dual information	3 days	3 days

Table 6 Requirement's estimation - Training companies.

Defect fix times (3 days)

As we can see in Table 7 it shows the bugs that the specified module had. In order to analyze the quantity of bugs 3 functionalities were analyzed.

Work Items	Number of Bugs
Formados Software - Operations	**3**
Tutor assignation to a specific area of the company	2
Planification of the learning and the project that the student will be assigned	1
Tracing of the dual formation process	0
Students' evaluation	0

Table 7 Errors and Bugs of the second case of study selected – Operations.

Work Items	Number of Bugs
Formados Software – Curriculum	**2**
Experience	1
Languages	1
Third level education	0
Aptitudes	0

Table 8 Error and bugs – Curriculum.

Work Items	Number of Bugs
Formados Software – Training Companies	**1**
General Information	0
Contact Information	0
Legal representative	1
Offices – Addresses	0
Tutors	0
Areas	0
Dual information	0

Table 9 Error and bugs – training companies.

Total regression test time (2 days)

There were implemented automated test for the following process: training companies and curriculum.

The automated curriculum test tested the most important functionalities of the process.

The Table 10 shows the test that this case of study has.

Kind of test	Number
Unit	8
Procedure	4
Automated	8
Total	20

Table 10 Test After CI Implementation

The Figure 30 show the automated test code for delete a register that was created by the test itself, it shows the code for click the delete button and the confirmation pop up.

The Figure 31 shows the result of the automated test.

```
@Given("Se va a eliminar el registro creado")
public void se_va_a_eliminar_el_registro_creado() {
    String idElementoEliminar = "form-experiencia-laboral:j_idt383:" + (numeroFilas) + ":btn-eliminar-exp";
    System.out.println("btn del registro que se va a eliminar:"+idElementoEliminar);
    WebElement element = driver.findElement(By.id(idElementoEliminar));
    Actions actions = new Actions(driver);
    actions.moveToElement(element).click().perform();
}

@When("Se confirma que se va a eliminar")
public void se_confirma_que_se_va_a_eliminar() {
    WebElement element = driver.findElement(By.id("form-eliminar-exp:btn-confirmacion-eliminar"));
    Actions actions = new Actions(driver);
    actions.moveToElement(element).click().perform();
}
```

Figure 30 Automated test code – Curriculum.

```
Scenario: Test Agregar Experiencia Laboral                    [90m# ec/forma
Starting ChromeDriver 83.0.4103.39 (ccbf011cb2d2b19b506d844400483861342c20cd-
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for sugg
ChromeDriver was started successfully.
ChromeDriver: chrome on MAC (abe74165a89bc06f5744ff9961c374bb)
logeoooooooooooooo
  [32mGiven [0m[32mClick  en hoja de Vida para abrir el Pop up[0m
  [32mWhen [0m[32mClick en el tab de expeiencia laboral[0m
------------------------------Experiencia Laboral
  [32mThen [0m[32mEl nombre en la tab debe ser [0m[32m[1m"Experiencia Laboral
  [32mGiven [0m[32mSe proporcionan datos de una nueva experiencia laboral[0m
fila del nuevo registro:0
  [32mThen [0m[32mSe verifican los nuevos datos ingresados[0m
  [32mGiven [0m[32mSe va a editar el registro creado[0m
  [32mWhen [0m[32mSe cambiala informacion del registro seleccionado[0m
  [32mThen [0m[32mSe verifica que se a editado correctamente el registro[0m
  [32mGiven [0m[32mSe va a eliminar el registro creado[0m
  [32mWhen [0m[32mSe confirma que se va a eliminar[0m
  [32mThen [0m[32mSe verifica que se elimino el registro[0m

1 Scenarios ([32m1 passed[0m)
11 Steps ([32m11 passed[0m)
0m22.098s
```

Figure 31 Automated test results.

Number of branches in version control

Number of branches: 2. As this project is starting, they just have 2 different big process to release to production.

This project has the number of branches depending on the module or a significant change.

Release to production (1 hour)

The software is released to production manually, but the DevOps have the stages of the pipeline that gave the results of the automate test and quality assurance.

In order to release a module or the entire system to production the company added new steps, all stage of the pipeline sent an email to all DevOps on the case of study in order to release or not to production, the steps are the following:

- Generate the ear with the new features.
- Review results of the built with maven.
- Review of quality assurance of the code.
- Review the results of automated test.
- Generate the ear with the new features.
- Update the database with all new tables and columns. The scripts are executed on the cloud database.
- Stop the wildfly server on the cloud and replace the ear file that is inside the deployments folder.
- Start the wildfly server with the new ear file.

This time to production is the first time that the software was released to production. In the future, each time that the software needs an update the developers team takes 1 hour.

Nonfunctional requirements result

These results show the response of the software to a determinate number of petitions and JMeter was used to gather the results.

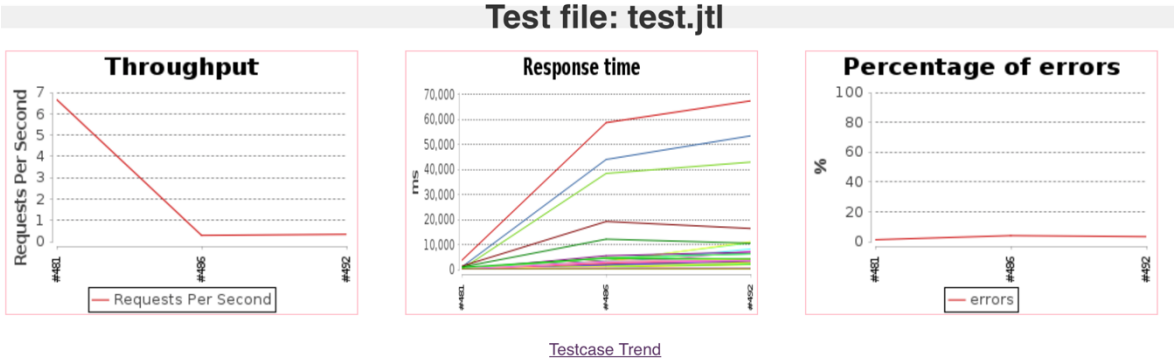

Figure 32 Nonfunctional requirements result.

6.4. CI Adoption and Implementation

Saviasoft uses an agile methodology that has their own pashes that the company has adopted with the years. The actual methodology and state are mentioned on the chapter (Chapter 3.3).

The implementation of the CI tools on the company took some time to the developers to get used to use them.

The time that took for the developers depended on each the project that they were assigned. The CI tools were implemented in all projects that the company were developing and the estimated time for the company to adopt and implement CI were 2 months.

In the estimated time was considered the time of the trainings and the implementation of the CI tools. Not all the CI tools were implemented on all projects, depending on each project there were implemented a determinate CI tool. The case study had all the CI tools.

On the Table 11 Projects Implemented we could see the time that took for each project.

Project	**Estimated Time**	**CI implemented (%)**
Formados (Case of study)	5 days	100%
Quiport	3 days	70%
A-vender	2 days	50 %
ILC	4 days	90 %
Quiport – AirSide	4 days	90%

Table 11 Projects Implemented.

There was a transition period before and after the CI implementation. The difference between both were detailed with each metrics of software before and after.

On the transition period it took place all trainings and courses about CI implementation and CI tools.

6.5. Questionnaire about CI

At the beginning the developers of the company filled out a Questionnaire in order to know how much the developers know about CI. The topics of the trainings that will be done will depend on the results of the questionnaire. This questionnaire: was made available in google surveys; and the time for answer was unlimited.

After the Questionnaire the answers were discussed between the developers to know the correct approach of each question.

The aim of the questionnaire is to know what the developers know about CI. The information collected by this survey is important because it will give the real state of the developers against CI and all answers of the developers will be analyzed and described. Depending on the answers the courses will be planned, other goal of the questionnaire is to give the same level of knowledge to all developers.

The answers to these questions (Figure 33) show that the developers know what continuous integration is, but some of them think that automated test is continuous integration. The approach of the developers could be because they had a little course about automated test.

This question was analyzed between all developers and all doubts were solved.

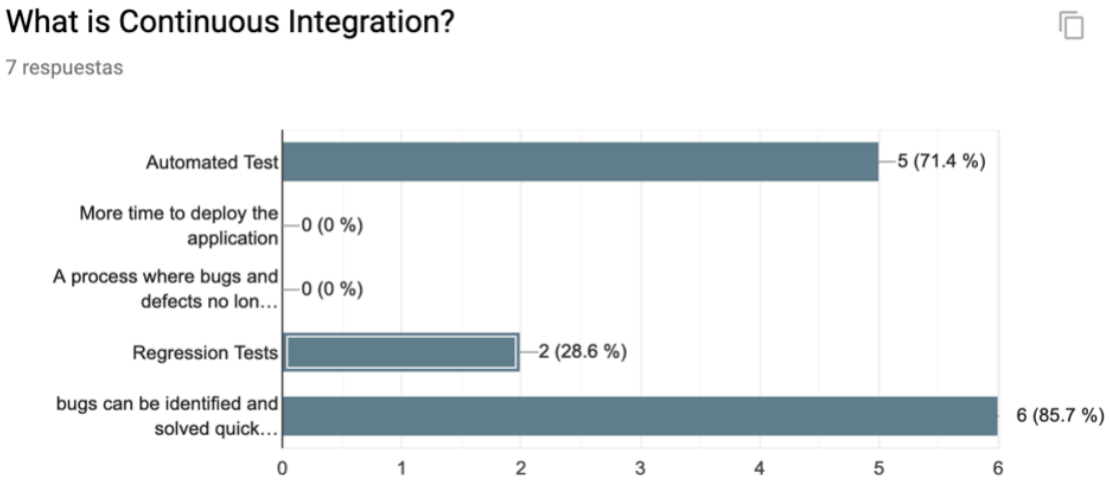

Figure 33 Results from Question: What is Continuous Integration?

All developers think that CI will help them (Figure 34), so we expect a good acceptation to introduce the new methodology. The developers are a little bit worried about the implementation because they do not know how it really works.

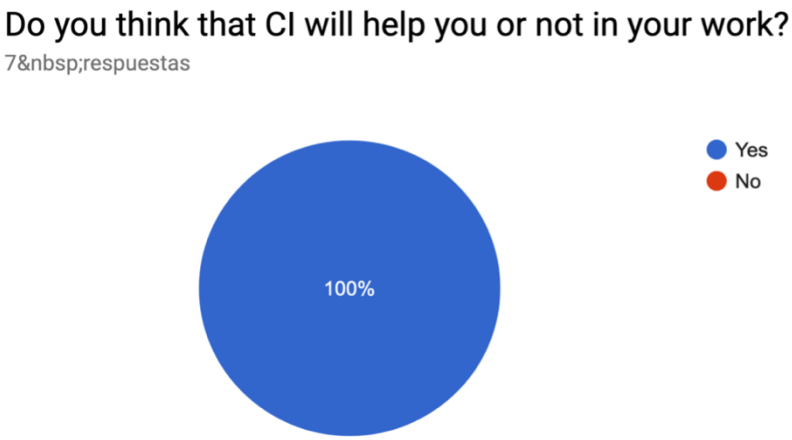

Figure 34 Results from Question: Do you think that CI will help you or not in your work?

Half of the developers have not used tools of CI, so they will take some time for them to get use to use this kind of tools (Figure 35).

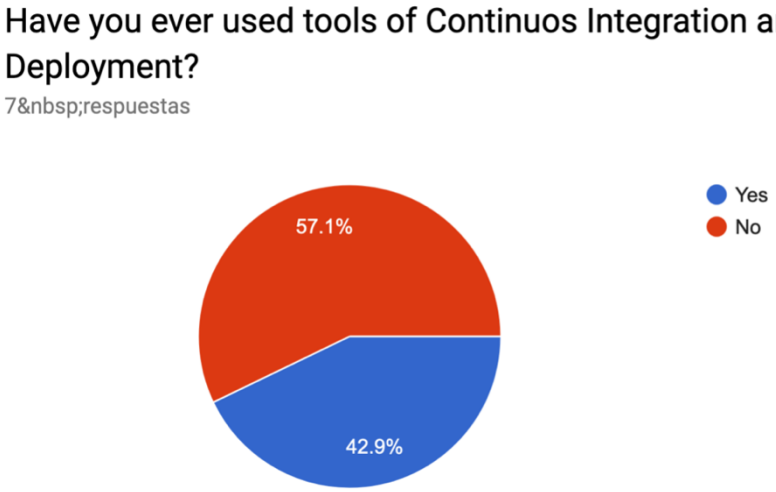

Figure 35 Results from the Question: Have you ever used tools of Continuous Integration and Continuous Deployment?

Most of the developers think that the implementation of CI will take a long time for them, therefore they do not know most of the tools and the features that CI has. It will take time to the developers to test and learn the DevOps tools that the company will use (Figure 36).

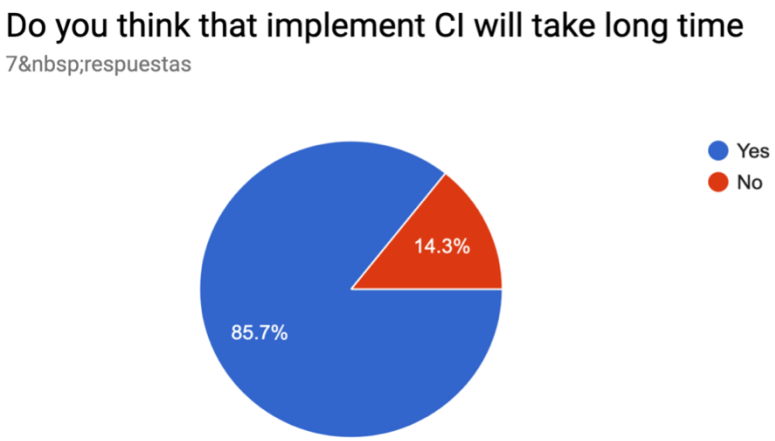

Figure 36 Results from the Question: Do you think that implement CI will take long time?

Most of the developers understand the main purpose of CI, there were just a few of them that thought some wrong issues about CI. Each answer to this question were analyzed and explained why it is the purpose or not (Figure 37).

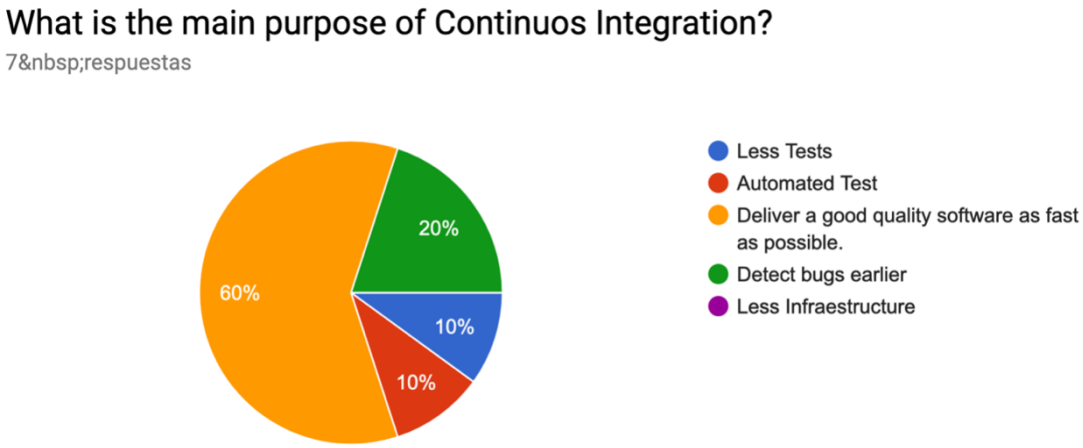

Figure 37 Results from the Question: What is the main purpose of Continuous Integration?

6.6. Questionnaire after CI

After the CI implementation the company developers answered a questionnaire in order to know the knowledge and experience acquired after the first implementation.

On Figure 38 we could realize that the developers understood very well CI concepts.

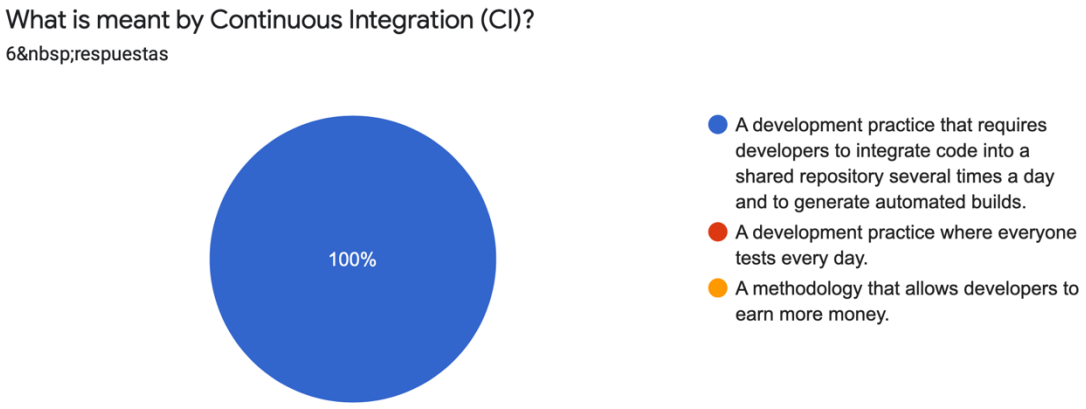

Figure 38 Results of Question: What is meant by Continuous Integration?

The Figure 39 show that the developers learn about CI, they know that the fact of apply its tools helps the software company. They know the success factors to accomplish a better software with CI.

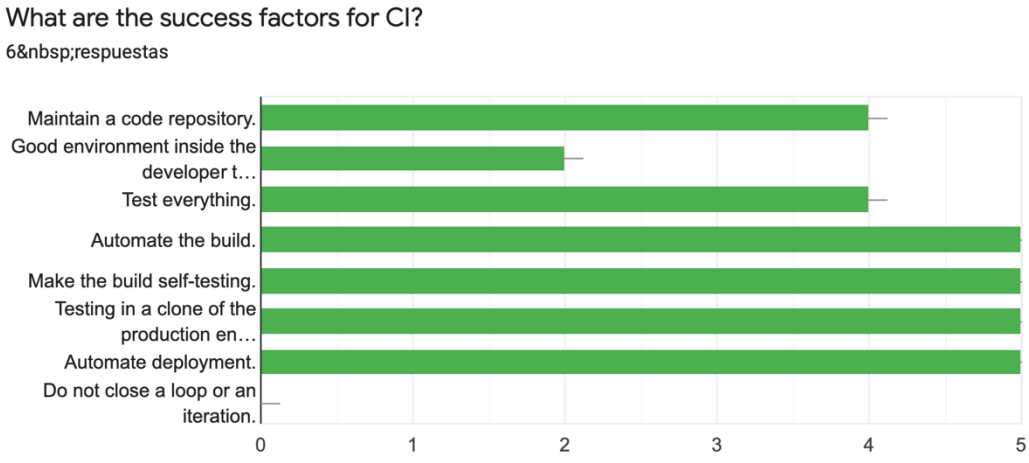

Figure 39 Results of Question: What are the sucess factors for CI?

The Figure 40 shows that most of the developers know the principal benefit of Continuous Testing because they have been using the new approach of testing that allows them to identify faster the bugs.

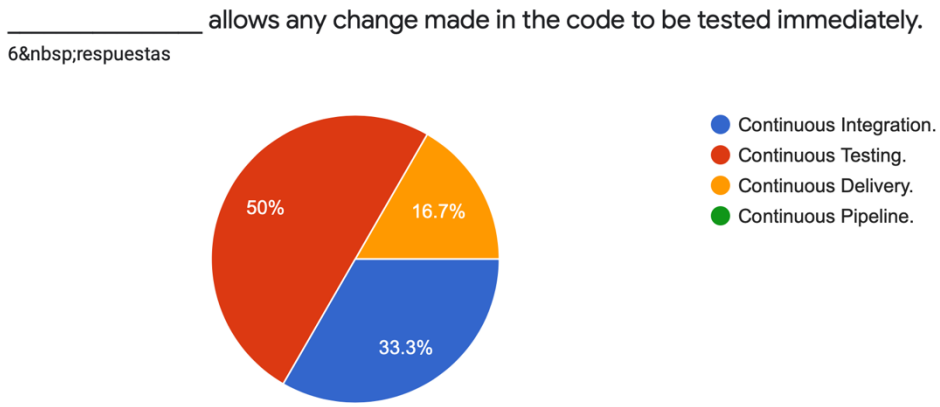

Figure 40 Results of Question that refers to Continuous Testing

Summary the overall results, the outputs of the Questionnaire after CI implementation show that the developers learn the CI concepts, they got used to use the CI tools implementing them on their projects.

6.7. Discussion of Results – The Impact of CI

Lead time per work item to production

The time that each item was released to production decreased considerably. Because of the automated test and the feedback, the developers could resolve the problems faster. When the developers do a change or develop a new module it is faster to release to production because of the automated release.

Although the implementation of the new CI tools took time to perform them, it is worth to have them because most of the metrics showed less time.

Defect fix times

The number of bugs [38] found decreased, but they were solved as soon as there were found.

In order to show the changes, the result of two modules were cited on this section.

This metrics shown better results because the developers team has SonarQube, which gives a feedback as soon as each commit is done. The developers changed the code as soon as they have the feedback of SonarQube that tells them that a determinate code could produce an error or does not follow code standards. That is the reason why the defects decrease, because they realize what will cause an error before producing.

Automated test

The automated test results were excellent, since not only the bugs decrease but also the develop time. On Figure 29 the results of a scenario show that there were no errors.

Identify the bugs took long time before the automated test. The automated test helps the developer's team to identify the errors and bugs of the software, consequently they could identify the bugs faster

Nonfunctional requirements

Now the developers team have statistics (Figure 32) of the server, allowing them to have a feedback that tells how the server reacts to determinate number of petitions on each process. They did not have this kind of information before CI Implementation.

Release to production

Each time that the developers do a push the pipeline executes automatically, and the developers control if the push will release to production or no.

The release to production time decreases because of the docker implementation, automated test and non-functional requirement test. This process was initially manually, and it is automatically now. As the process is automatically each time that the software requires an update it takes 5 or 10 minutes maximum, that is the time that the pipeline takes long.

Analysis

CI was successfully implemented in the software company, there was a transition period before and after CI implementation, this period was detailed on the chapter (6.4).

The results of the CI Implementation are good, they show that the company has a better software quality, a better test process, and better statistics to analyze the implementation of each software.

The results now are visible, not only for the developers but also to the clients, because they know that the company has CI on their development process.

Challenges

In order to implement CI on the methodology of Saviasoft, there were some challenges [39] to manage:

- Learn CI methodology.
- Get used to use the CI tools.
- Implement CI tools.

The first challenge was teaching to the developers the CI concepts and tools. They were skeptical to implement all CI tools on their development process. They taught that it would take long time to get use to the new process.

The implementation of each tool that is cited on this dissertation took a considerably time the first time that it was implemented. When the process became easier the implementation was faster.

Summing up, the impact [40] that CI had on the company was good. CI changed the way of developing the software.

The results of CI were noticed as soon as each tool was implemented, because the developers realize that they have a better feedback, and they have better results to analyze.

7. Conclusions and Future Work

As we can realize comparing the estimation and number of bugs, the estimation of each item was the same of less, and the bugs found per requirement were not a lot, as conclusion the time of developing and the quality of software is better. The time for resolving bugs and the time to launch and deliver the software decreased consequently the Quality increased thank to the CI tools implemented.

It was difficult for the developers spend more time for the CI implementation on their software projects, because at the beginning it requires more time but at the end it will save time that would have taken to resolve errors or problems. Though the CI implementation had their complication and challenges all developers learn the concepts and the tools and they realized that although it takes an extra time to implement in the end it helps them to have better software quality and better feedback.

The CI implementation on the company was successfully implemented. The company developers were a little bit worried at the beginning because they taught that the CI would take more time for them, in fact it takes some time, but it gives them big advantages that help not only the developers, but also the company.

The implementation of CI was a challenge in the company, although it helps to the software development process. The CEOs of the company are happy with the CI implementation because they know that the company will produce a software with a better quality. The CI implementation took some time to the developers to get used to it. The developers end up using CI tools as a standard and they realize that it helps them on their software projects.

Implementing CI on a software company that did not have was exiting, because the changes and results before and after are remarkable, the CEOs of the company, developers, clients realize the change and the result after the implementation.

Carry out training to the colleagues was interesting, because teach a new approach of develop with CI changed their way of thinking about software.

The automatic release of the software is being done; it will be implemented slowly because the company decide to implement in a second phase of the CI implementation

Bibliography

[1] M. Shahin, M. Ali Babar, and L. Zhu, “Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices,” *IEEE Access*, vol. 5, no. Ci, pp. 3909–3943, 2017, doi: 10.1109/ACCESS.2017.2685629.

[2] B. Fitzgerald and K.-J. Stol, “The Journal of Systems and Software Continuous software engineering: A roadmap and agenda,” *J. Syst. Softw.*, vol. 19, pp. 1–14, 2015, doi: 10.1016/j.jss.2015.06.063.

[3] G. A. Campbell and P. P. Papapetrou, *SonarQube in action*. Greenwich, Connecticut, USA: Manning Pub, 2014.

[4] W. Van Casteren, “The Waterfall Model And The Agile Methodologies : A Comparison By Project Characteristics-Short The Waterfall Model and Agile Methodologies,” *Acad. Competences Bachelor*, no. February, pp. 10–13, 2017, doi: 10.13140/RG.2.2.10021.50403.

[5] B. W. Boehm, “A Spiral Model of Software Development and Enhancement,” *Computer*, vol. 21, no. 5. pp. 61–72, 1988, doi: 10.1109/2.59.

[6] P. Beynon-Davies, C. Came, H. Mackay, and D. Tudhope, “Rapid application development (Rad): An empirical review,” *Eur. J. Inf. Syst.*, vol. 8, no. 3, pp. 211–232, 1999, doi: 10.1057/palgrave.ejis.3000325.

[7] K. Beck, *Extreme Programming Explained , Second Edition*. 2004.

[8] S. Sharma, D. Sarkar, and D. Gupta, “Agile Processes and Methodologies: A Conceptual Study Sheetal,” *Int. J. Comput. Sci. Eng.*, vol. 4, no. 05, pp. 892–898, 2012.

[9] T. Dingsøyr, “PREPRINT: 1 Agile Development at Scale: The Next Frontier,” no. April, pp. 1–10, 2019.

[10] M. Sameen Mirza and S. Datta, “Strengths and Weakness of Traditional and Agile Processes - A Systematic Review,” *J. Softw.*, vol. 14, no. 5, pp. 209–219, 2019, doi:

10.17706/jsw.14.5.209-219.

[11] "Agile Model - javatpoint." https://www.javatpoint.com/software-engineering-agile-model (accessed Apr. 13, 2020).

[12] P. Adi, "Scrum Method Implementation in a Software Development Project Management," *Int. J. Adv. Comput. Sci. Appl.*, vol. 6, no. 9, pp. 198–204, 2015, doi: 10.14569/ijacsa.2015.060927.

[13] H. Demaio, "Getting started with PKI," *Inf. Syst. Secur.*, vol. 8, no. 2, pp. 27–36, 1999, doi: 10.1201/1086/43305.8.2.19990601/31063.7.

[14] "Continuous integration | ThoughtWorks." https://www.thoughtworks.com/es/continuous-integration (accessed Nov. 04, 2019).

[15] H. Jez and D. Farley, *Continues Delivery*. Martin Fowler.

[16] M. Zanoni, F. Perin, F. A. Fontana, and G. Viscusi, "A Qualitative Study of DevOps Usage in Practice," *J. Softw. Evol. Process*, vol. 26, no. 12, pp. 1172–1192, 2014, doi: 10.1002/smr.

[17] Y. Gupta, *Software Quality Assurance*, vol. 6, no. 4. 1989.

[18] N. Noon, "DevOps Image." https://hackernoon.com/hn-images/1*t4ufHLq4izwXV6PoMkYgWw.jpeg (accessed Apr. 11, 2020).

[19] V. Fedak, "DevOps Team Roles And Responsibilities - By," 2018. https://hackernoon.com/devops-team-roles-and-responsibilities-6571cfb56843 (accessed Nov. 17, 2019).

[20] Www.blazemeter.com, "CONTINUOUS TESTING IN PRACTICE Completing the Continuous Delivery Process."

[21] "Running Non-Functional Tests in Continuous Testing Mode – Part 2 | Zuci Systems." https://www.zucisystems.com/running-non-functional-tests-in-continuous-testing-mode-continued/ (accessed Nov. 17, 2019).

[22] "Functional Testing Vs Non-Functional Testing: What's the Difference?" https://www.guru99.com/functional-testing-vs-non-functional-testing.html (accessed Apr. 11, 2020).

[23] "Non-Functional Testing." https://newline.tech/blog/non-functional-testing/ (accessed Apr. 11, 2020).

[24] S. Amber, "Strategies for Verifying Non-Functional Requirements – Disciplined Agile (DA)," 2018. https://disciplinedagiledelivery.com/strategies-for-verifying-non-functional-requirements/ (accessed Nov. 17, 2019).

[25] "Security Testing,Penetration & Vulnerability Testing Services & Tools." http://www.odooqa.com/page/security-testing (accessed Apr. 15, 2020).

[26] "Evaluation results - Usability testing - 6203 digital library evaluation." http://6203digitallibrary.weebly.com/evaluation-results---usability-testing.html (accessed Apr. 11, 2020).

[27] "Code Quality and Security | SonarQube." https://www.sonarqube.org/ (accessed Apr. 11, 2020).

[28] "Katalon Automation Recorder - Powerful Selenium IDE to record, debug, play tests in any browser." https://www.katalon.com/resources-center/blog/katalon-automation-recorder/ (accessed Apr. 11, 2020).

[29] A. Häkli, "Implementation of Continuous Delivery Systems," no. May, pp. 1–62, 2016, doi: 10.13140/RG.2.2.33882.59846.

[30] S. Kejadiwal, "Continuous Integration Using Jenkins."

[31] B. Bashari Rad, H. John Bhatti, and M. Ahmadi, "An Introduction to Docker and Analysis of its Performance," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 3, pp. 228–235, 2017.

[32] Tutorialspoint, "Cucumber Tutorial," 2016.

[33] "Using Selenium and WebDriver | Electron." https://www.electronjs.org/docs/tutorial/using-selenium-and-webdriver (accessed Jul. 06, 2020).

[34] S. Shenoy, N. A. A. Bakar, and R. Swamy, "An adaptive framework for web services testing automation using JMeter," *Proc. - IEEE 7th Int. Conf. Serv. Comput. Appl. SOCA 2014*, no. January, pp. 314–318, 2014, doi:

10.1109/SOCA.2014.36.

[35] N. Naseer, “Advantages of using JMeter for non-functional automation testing.” https://suyati.com/blog/advantages-of-using-jmeter-for-non-functional-automation-testing/ (accessed Jul. 06, 2020).

[36] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, “Software testing techniques: A literature review,” *Proc. - 6th Int. Conf. Inf. Commun. Technol. Muslim World, ICT4M 2016*, no. November, pp. 177–182, 2017, doi: 10.1109/ICT4M.2016.40.

[37] “How to Create a Gantt Chart.” https://www.gantt.com/creating-gantt-charts.htm (accessed May 29, 2020).

[38] E. Murphy-Hill, T. Zimmermann, C. Bird, and N. Nagappan, “The design of bug fixes,” *Proc. - Int. Conf. Softw. Eng.*, pp. 332–341, 2013, doi: 10.1109/ICSE.2013.6606579.

[39] “Challenges of Implementing Continuous Integration (CI) at Scale.” https://www.cigniti.com/blog/challenges-of-implementing-continuous-integration-testing/ (accessed May 28, 2020).

[40] A. Bhattacharya, “Impact of Continuous Integration on Software Quality and Productivity.” 2014, Accessed: May 29, 2020. [Online]. Available: http://rave.ohiolink.edu/etdc/view?acc_num=osu1410945557.

Appendices

Appendix A - Gantt Diagram of the dissertation

	Name	Duration	Start	Finish	Predecess...
	Master Thesis	156 days	**11/18/19 8:00 AM**	**6/22/20 5:00 PM**	
	Project proposal	10 days	11/18/19 8:00 AM	11/29/19 5:00 PM	
	G1: Research: How to measure software before and after CI	30 days	12/2/19 8:00 AM	1/10/20 5:00 PM	2
	G2: Set up steps for implement CI	39.625 days	**1/13/20 11:00 AM**	**3/6/20 5:00 PM**	
	Couses	17.625 days	**1/13/20 11:00 AM**	**2/5/20 5:00 PM**	
	Introduction of CI	2 days	1/13/20 11:00 AM	1/15/20 11:00 AM	3
	My first pipeline	3 days	1/16/20 8:00 AM	1/20/20 5:00 PM	6
	Automated test	6 days	1/21/20 8:00 AM	1/28/20 5:00 PM	7
	SonarQube	3 days	1/29/20 8:00 AM	1/31/20 5:00 PM	8
	Jmeter	3 days	2/3/20 8:00 AM	2/5/20 5:00 PM	9
	Select the measures	5 days	2/6/20 8:00 AM	2/12/20 5:00 PM	10
	Analyze the Saviasoft methodology	8 days	2/26/20 8:00 AM	3/6/20 5:00 PM	11
	G3: Measure the software results before implementing a continuous integration methodology	17 days	**2/13/20 8:00 AM**	**3/6/20 5:00 PM**	
	Select a base line project	2 days	2/13/20 8:00 AM	2/14/20 5:00 PM	12
	Measure the results with the current methodology	15 days	2/17/20 8:00 AM	3/6/20 5:00 PM	14
	G4: Implement the CI methodology	51 days	**3/9/20 8:00 AM**	**5/18/20 5:00 PM**	
	Select a base line project	2 days	3/9/20 8:00 AM	3/10/20 5:00 PM	15
	Jenkins installation	2 days	3/11/20 8:00 AM	3/12/20 5:00 PM	17
	Create the pipeline	2 days	3/13/20 8:00 AM	3/16/20 5:00 PM	18
	Create webhooks linked to git repository	2 days	3/17/20 8:00 AM	3/18/20 5:00 PM	19
	Conect the git repo to the jenkins project	2 days	3/19/20 8:00 AM	3/20/20 5:00 PM	20
	Configure the SonarQube	2 days	3/23/20 8:00 AM	3/24/20 5:00 PM	21
	Configure a Jmeter	2 days	3/25/20 8:00 AM	3/26/20 5:00 PM	22
	Manual Test	2 days	3/27/20 8:00 AM	3/30/20 5:00 PM	23
	Do the automated test	15 days	3/31/20 8:00 AM	4/20/20 5:00 PM	24
	Create the automate deployment	10 days	4/21/20 8:00 AM	5/4/20 5:00 PM	25
	Create the automate release	10 days	5/5/20 8:00 AM	5/18/20 5:00 PM	26
	G5: Measure the result of the new methodology	15 days	**5/19/20 8:00 AM**	**6/8/20 5:00 PM**	
	Measure the same issues that were analized with the G3	15 days	5/19/20 8:00 AM	6/8/20 5:00 PM	27
	G6: Compare G3 and G5	10 days	**6/9/20 8:00 AM**	**6/22/20 5:00 PM**	
	Compare each issue of both projects	10 days	6/9/20 8:00 AM	6/22/20 5:00 PM	29

Figure 1 Activities of the dissertation

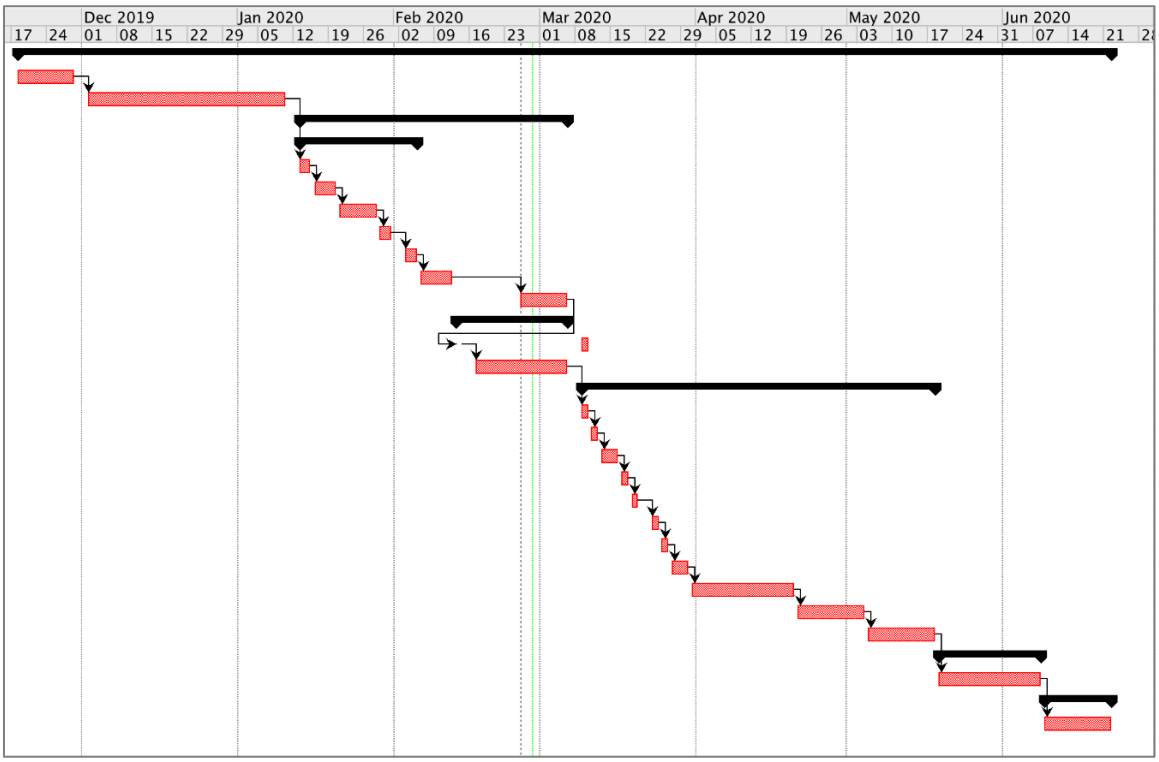

Figure 2 Timeline of the dissertation

Appendix B - Gant Diagram of the base line project

	Name	Duration	Start	Finish	Predecess...
	PROYECTO	33 days	**9/6/19 8:00 AM**	**10/22/19 5:00 PM**	
	Análisis (o puede ser explicación de procesos detallados, si el client	2 days	**9/6/19 8:00 AM**	**9/9/19 5:00 PM**	
	Levantamiento detallado de información	2 days	9/6/19 8:00 AM	9/9/19 5:00 PM	
	Desarrollo	24 days	**9/10/19 8:00 AM**	**10/11/19 5:00 PM**	
	Modulos del aplicativo	24 days	**9/10/19 8:00 AM**	**10/11/19 5:00 PM**	
	A-vender	18 days	**9/10/19 8:00 AM**	**10/3/19 5:00 PM**	
	Tablas dee base, persistencia en java y maquetacion de servicios.	2 days	9/10/19 8:00 AM	9/11/19 5:00 PM	3
	Servicios para guardar nuevos datos de liquidaciones	8 days	9/12/19 8:00 AM	9/23/19 5:00 PM	7
	Servicios web para recibir los datos desde el cliente.	8 days	9/24/19 8:00 AM	10/3/19 5:00 PM	8
	Conexión desde Actuaria	4 days	**10/4/19 8:00 AM**	**10/9/19 5:00 PM**	
	Servicios web para enviar los datos desde actuaría hacia a-vender.	4 days	10/4/19 8:00 AM	10/9/19 5:00 PM	9
	Reportes	2 days	**10/10/19 8:00 AM**	**10/11/19 5:00 PM**	
	Liquidacion de compras(pdf)	2 days	10/10/19 8:00 AM	10/11/19 5:00 PM	11
	Pruebas	3 days	**10/14/19 8:00 AM**	**10/16/19 5:00 PM**	
	Elaboración Documento de casos de prueba	0.5 days	10/14/19 8:00 AM	10/14/19 1:00 PM	13
	Revisión y aprobación de documento de casos de prueba	0.5 days	10/14/19 1:00 PM	10/14/19 5:00 PM	15
	Pruebas funcionales totales	1 day	10/15/19 8:00 AM	10/15/19 5:00 PM	16
	Mejoras y correcciones	1 day	10/16/19 8:00 AM	10/16/19 5:00 PM	17
	Puesta en producción	3 days	**10/17/19 8:00 AM**	**10/21/19 5:00 PM**	
	Cambios en jar de actuaria	1 day	10/17/19 8:00 AM	10/17/19 5:00 PM	18
	Puesta a produccion en a-vender	2 days	10/18/19 8:00 AM	10/21/19 5:00 PM	20
	Seguimiento post-producción (De ser el caso, en dias)	1 day	10/22/19 8:00 AM	10/22/19 5:00 PM	21

Figure 3 Activities of the base line project

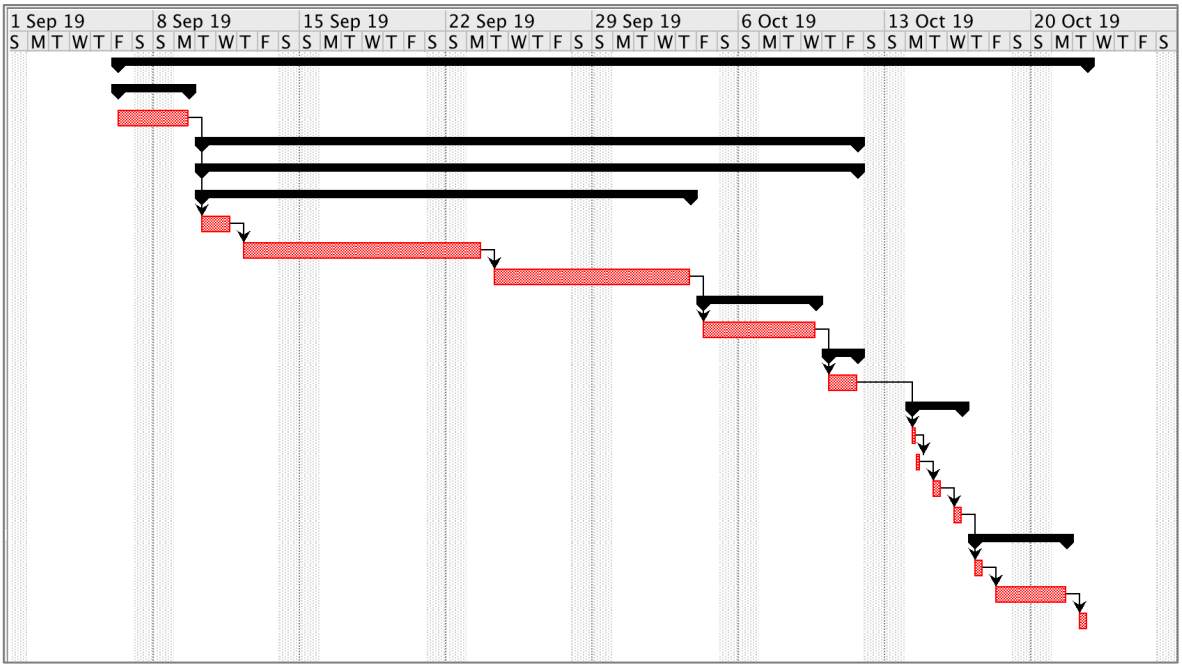

Figure 4 Timeline of the base line project

Appendix C - Course: Introduction to CI & CD

Figure 5 Slide 1 of the Course: Introduction to CI & CD

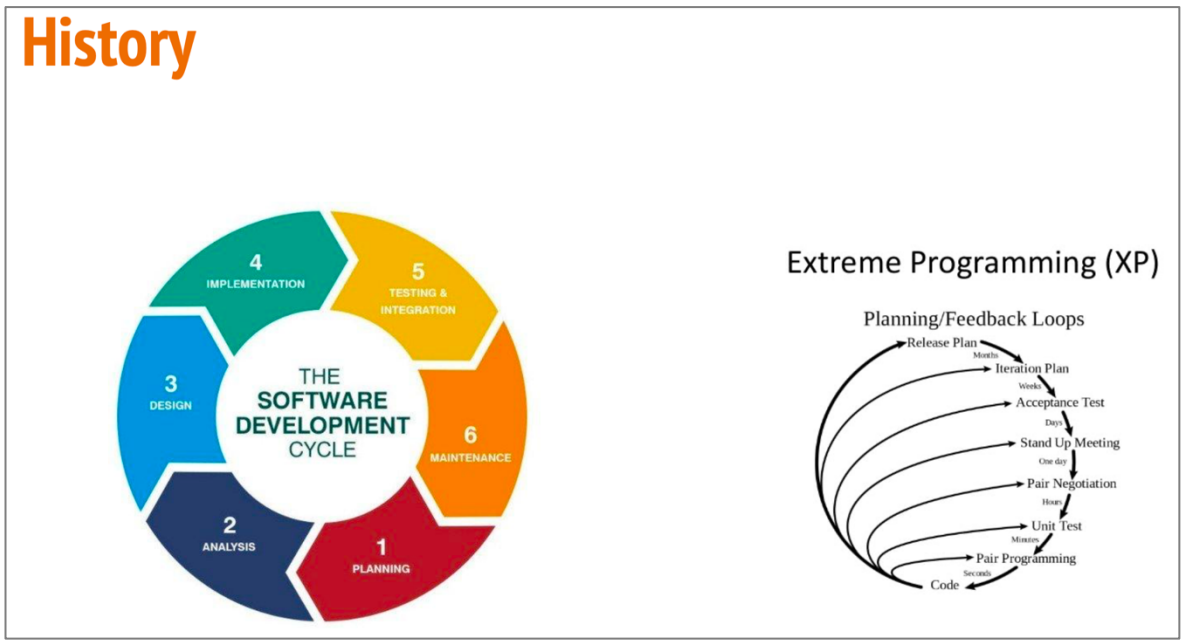

Figure 6 Slide 2 of the Course: Introduction to CI & CD

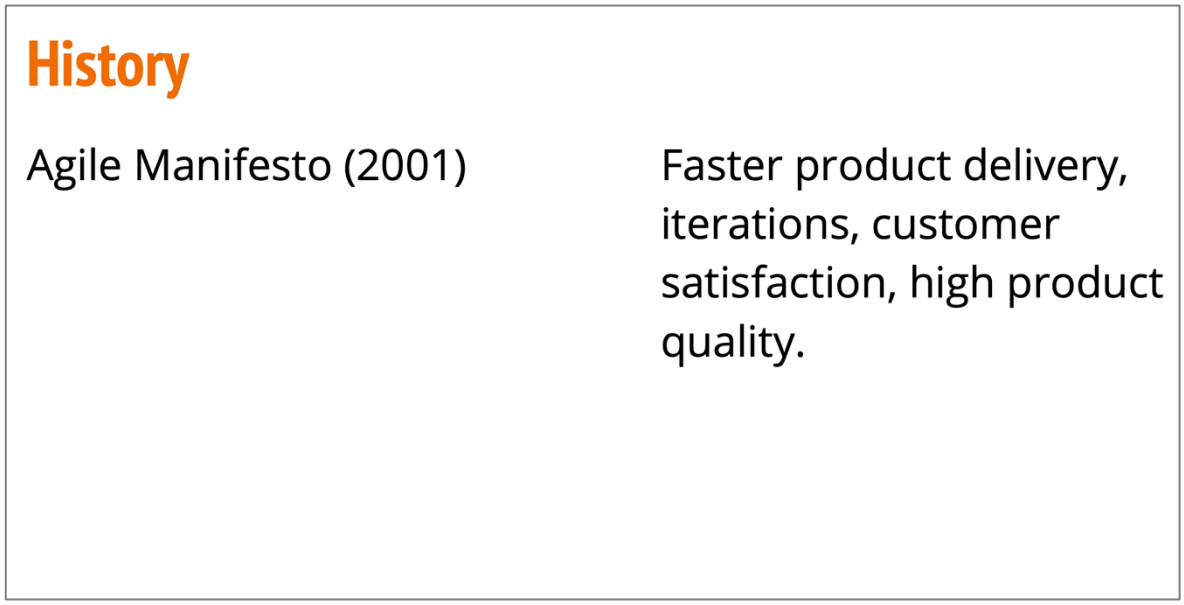

Figure 7 Slide 3 of the Course: Introduction to CI & CD

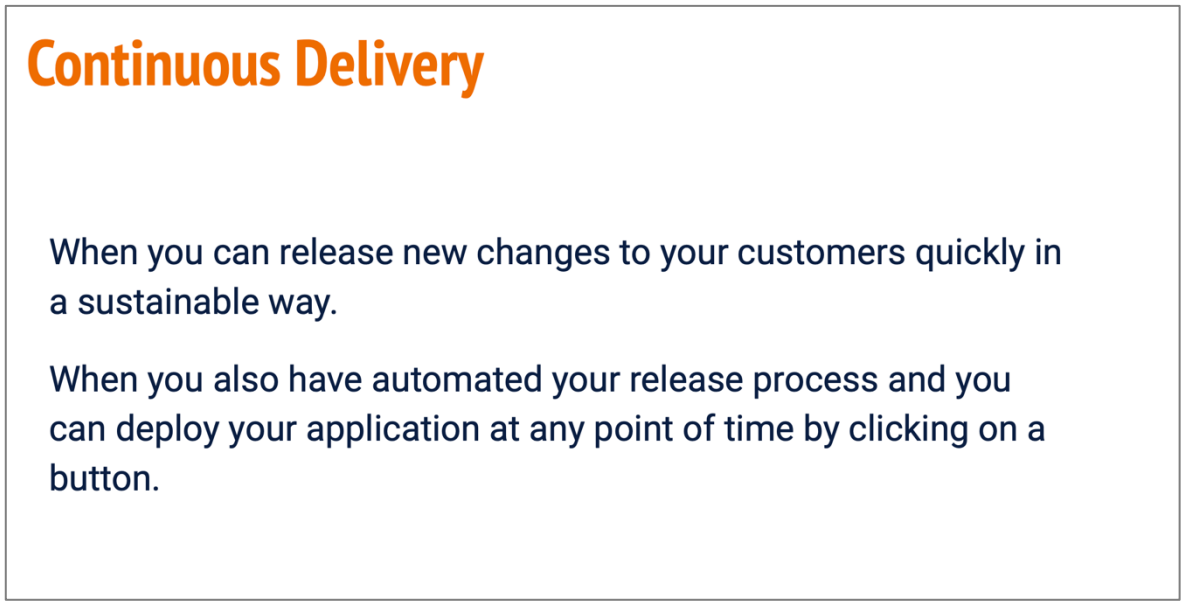

Figure 8 Slide 4 of the Course: Introduction to CI & CD

Figure 9 Slide 5 of the Course: Introduction to CI & CD

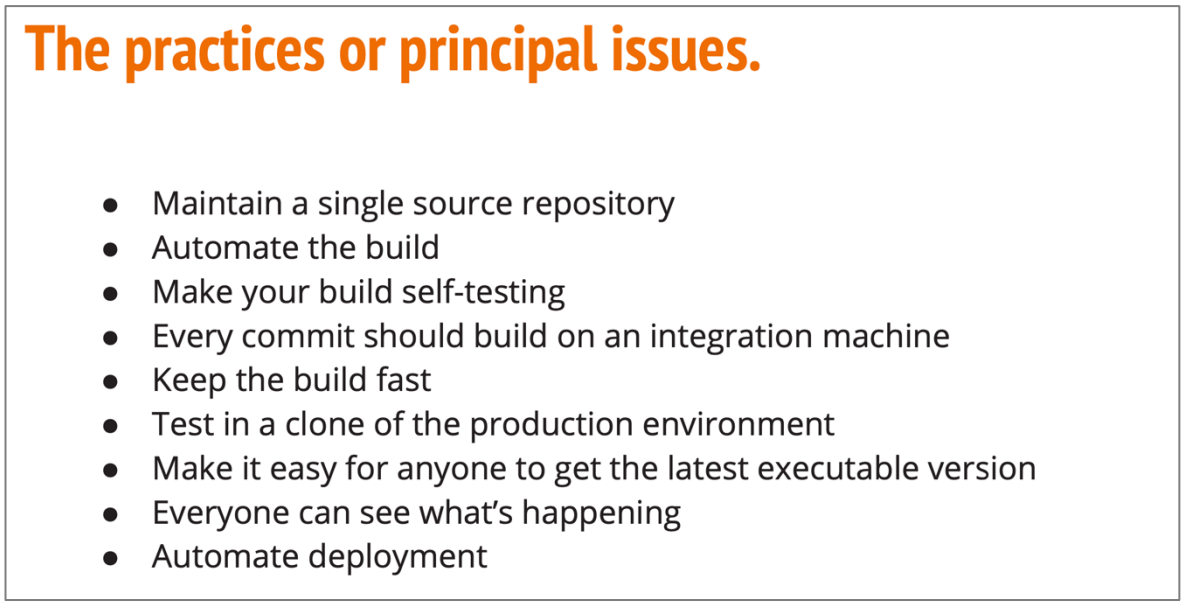

Figure 10 Slide 6 of the Course: Introduction to CI & CD

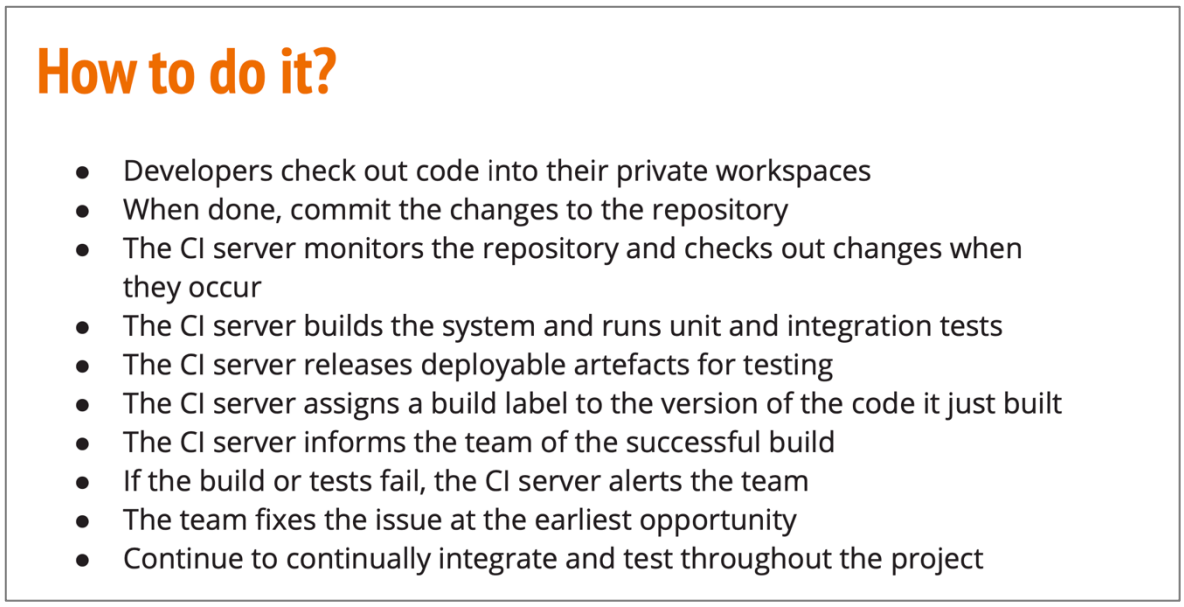

Figure 11 Slide 7 of the Course: Introduction to CI & CD

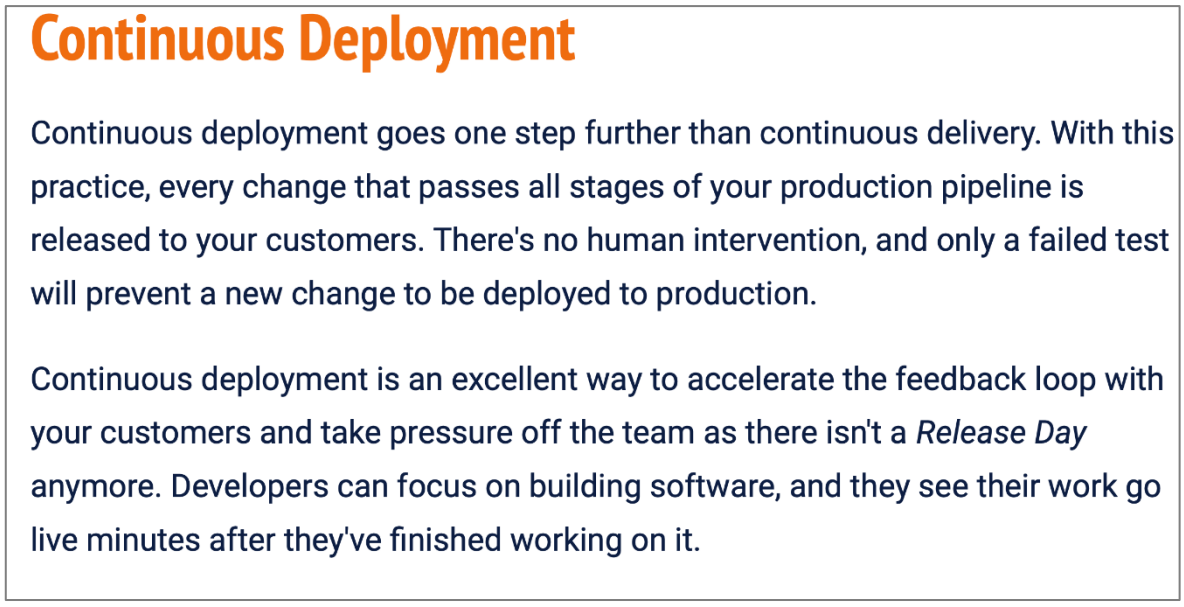

Figure 12 Slide 8 of the Course: Introduction to CI & CD

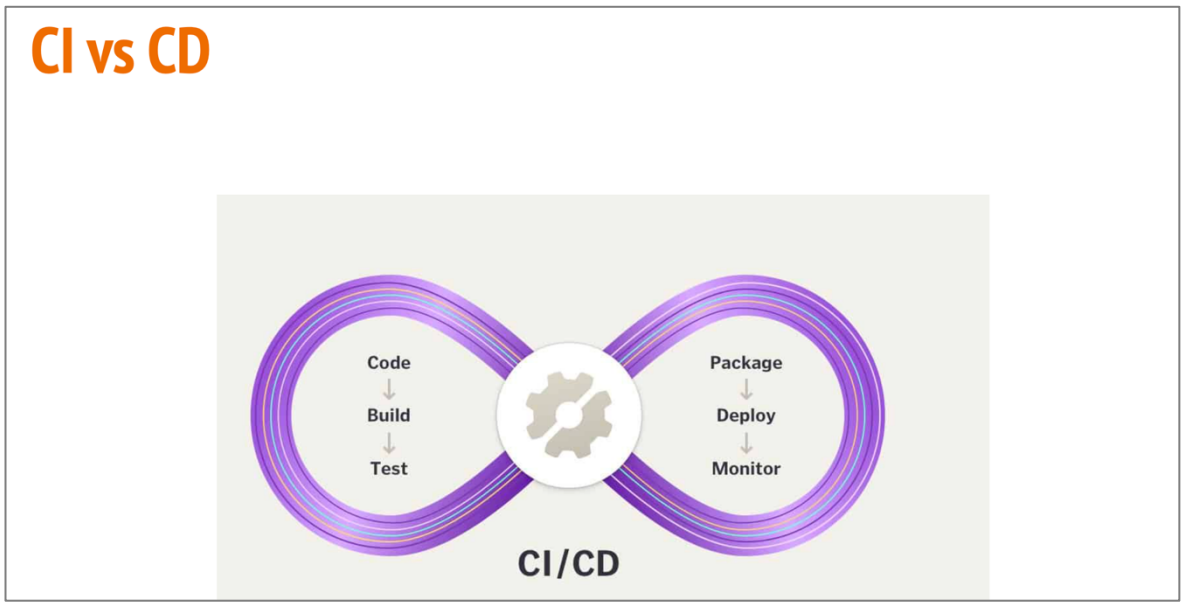

Figure 13 Slide 9 of the Course: Introduction to CI & CD

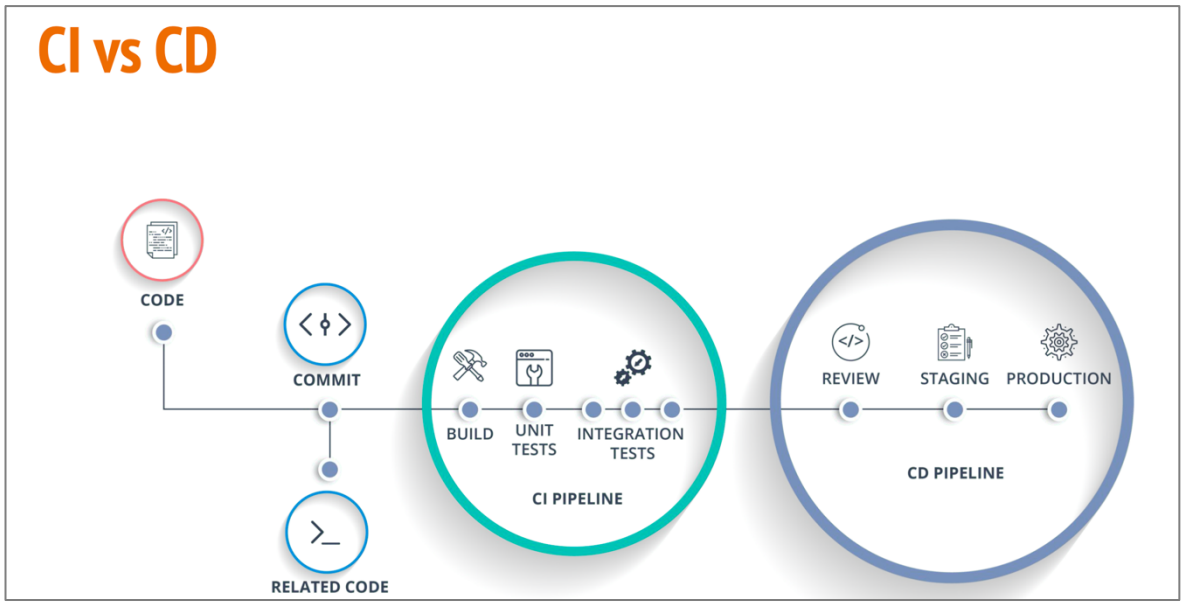

Figure 14 Slide 10 of the Course: Introduction to CI & CD

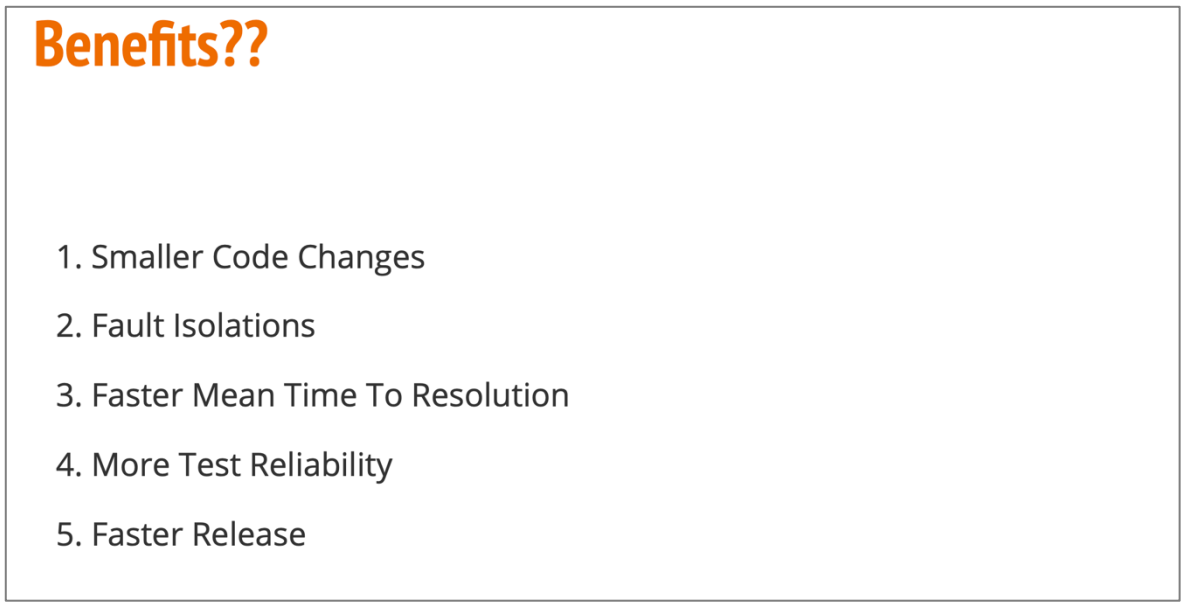

Figure 15 Slide 11 of the Course: Introduction to CI & CD

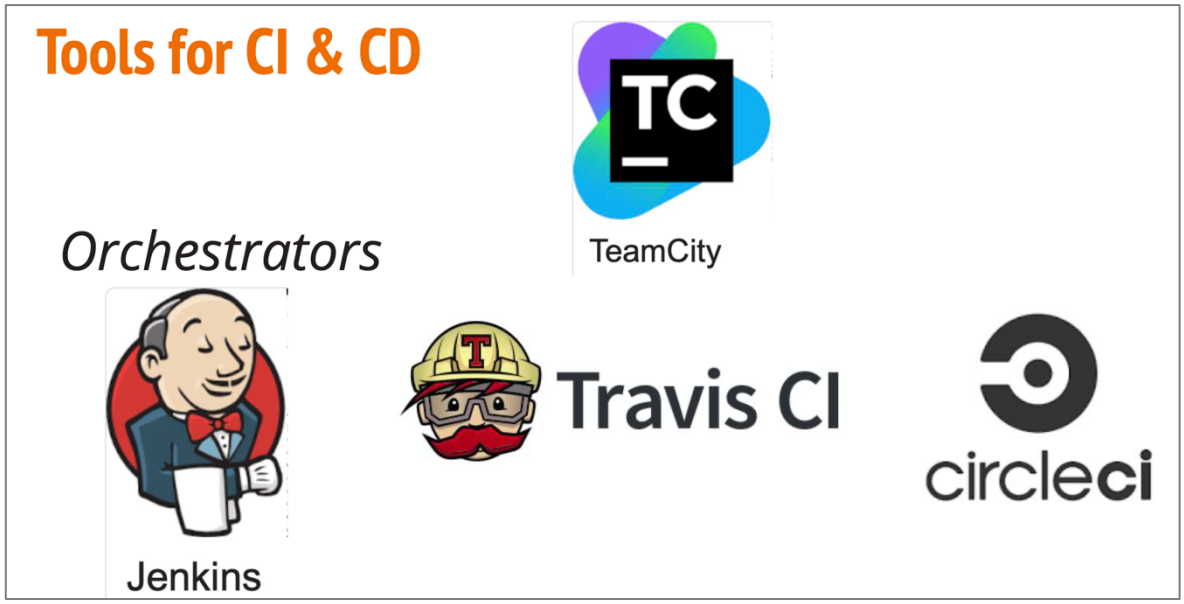

Figure 16 Slide 12 of the Course: Introduction to CI & CD