UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

INFORMATION ENGINEERING DEPARTMENT

MASTER'S DEGREE IN
ICT FOR INTERNET AND MULTIMEDIA

"A Study of Efficiency Improvement in Test Automation for Electronic Invoicing Software Solutions"

Supervisor: Leonardo Badia

Candidate: Bilge Ozcanbaz
2008489

ACADEMIC YEAR 2021 – 2022

Graduation Date: 05 December 2022

Thanks

First, I would like to thank my family, who supported me on my decision and my second family, Elif Altınbaş, Hilal Motçu and Beste İçten, for standing by me every minute despite the 2300 kilometers between us.

I would like to thanks to my roommate Elif Bakar for encouraging me when I feel lost and to Mattia Bolzonella for all the help with work and Italian life.

I would like to thank all the friends and fellow students who have been part of this adventure.

Furthermore, I would like to thank Fabio Canevarolo, my company tutor, who believed in me and gave me the opportunity to grow. And to Claudio Mazzuco for supporting and comforting me during the preparation of this thesis.

Finally, I would like to express my gratitude to Prof. Badia, supervisor of my thesis, for the help and support provided to me during the writing of the work.

Padova, December 2022 Bilge Ozcanbaz

Abstract

As with the growing technical industries, number of software system has been implemented as web-based applications. Some of these web applications are very complex and it is very difficult to test such complex web applications. Automation testing uses automation tools to reduce human intervention and repeatable tasks. In this paper we have designed and implemented automation testing framework for testing web applications. This new automation testing framework has been implemented using selenium RemoteWebDriver() tool instead of selenium WebDriver(). Using this framework tester can easily write their test cases efficiently and in less time. This framework is helpful to developer to analyze their code due to screen shot property of framework. In this thesis I wanted to show you that RemoteWebDriver() is very easy to maintain and repair the test suite for new release of the application using this framework. Due to the Results Section, RemoteWebDriver() has better performance and 3 seconds faster than WebDriver().

Keywords*: Test Improvement, Software Testing, Selenium IDE, Performance Evaluation*

Contents

List of Figures

List of Tables

1. Introduction

The aim of this project is to improve the software test performance by comparing WebDriver() and RemoteWebDriver(). Selenium WebDriver() is a tool that allows you to automatically create and operate some test steps of websites through your web browser. [1] And Selenium RemoteWebDriver() is used to execute the browser automation suite on a remote machine. In other words, RemoteWebDriver() is a class that implements the WebDriver() interface on the remote server. [1]

Even though there are other methods that might slow the test process, since RemoteWebDriver() is a class that implements a WebDriver() interface, it controls the node PC over Grid [2] which means RemoteWebDriver() separates where the tests are running from where the browser is and allows tests to be run with browsers that are not available on the current OS, it could be much more efficient and faster to run the tests.

The problem, quite simply, is that testing is difficult, and only gets more difficult as organizations look to drive automation at scale. Nowhere is this challenge more evident than in organizations' ongoing struggles with test quality. [3] Organizations must find ways to improve test quality and speed. A major reason so many tests fail is they take far too long to run. The longer it takes to run, the more likely it is to end with a failure. So, to ignore these failures we need to be faster while testing.

This brings my thesis logic: making the tests of Invoice Channel® project is faster and easier to implement. Invoice Channel®, is the software solution that automates the flow of electronic invoicing with a "zero impact" approach, as, thanks to web services, it allows an easy integration to third-party ERP systems [4].

Software testing is a critical stage of the software development life cycle. Today numerous software applications are written as a web-based application that operates on an Internet Browser. Software Testing is a method, which applies, running a software program/application and uncovering all errors or bugs in that program/application so that the result will be defect-free software. The quality of any software can only be known through means of testing (software testing). [5]

The digitalization of invoice operations provides an excellent opportunity for companies to lessen expenses, optimize organizational tasks, and improve efficiency and competitiveness.

Electronic Invoicing is a process regulated by Italian law that allows you to issue, receive, digitally store, and exhibit tax documents electronically, ensuring that the content is immutable and unalterable over time.

So, in this thesis I will explain firstly the current knowledge about WebDriver() and RemoteWebDriver() on Chapter 2, then on Chapter 3, the methodologies of the technologies which is used during the preparation of thesis, on Chapter 4 is the explanation of the theoretical part of the test and importance of the software testing and finally on the Chapter 5 description of project, the task, the results, and at the end conclusion.

2. State of the Art

In this section, I will explain what the problem is, why is it important and how the agile methodology relates to the problem and the theoretical part of the thesis as what Selenium WebDriver() is, Selenium RemoteWebDriver() and the differences between.

Software testing is a key component of software quality assurance and represents the ultimate verification of specifications, design, and coding. Software testing is the process of testing the functionality and correctness of software through execution. The importance and complexity of software testing is reflected by the costs involved: 30% to 80% of the development costs are reported to be related to testing and studies on release time indicate that most of the release time is consumed by testing. The cost and time involved in testing can be managed through test automation, where the execution of a test is performed by software instead of a human. [6]

In this aspect, software testing plays an important role in the software development lifecycle and is very good at identifying process problems. To identify the problems and fix them, the testing process must be as fast as it can be.

Whether some API calls are more energy intensive than others, and whether sequences of API calls (patterns) are repeated frequently, causing anomalies in energy consumption. In this study, we analyzed execution traces of 55 Android applications looking for the most performance-hungry Android API calls. Here are the main results of this work.

1. GUI, image manipulation, and database related APIs consume the most power.
2. Using getters and setters when accessing inner class fields is energy intensive. This result is consistent with previous work and constitutes a trade-off between information hiding and power efficiency.
3. Updating application views and widgets consumes a lot of energy. [7]

In many software applications, the timeliness of monitoring information plays an important role in system performance. The age of information is a performance metric describing the freshness of updates and is of particular interest for remote sensing scenarios involving a high number of nodes, as is expected to happen, e.g., in emergency scenarios or industrial applications of the Internet of Things. [8] Age of Information (AoI) is a key performance indicator in mission-critical and time-sensitive applications, including smart transportation, healthcare, remote surgery, robotics cooperation, public safety, industrial process automation, to count a few. [9]

Throughout the software development lifecycle, measurement processes should be used effectively to evaluate the quality, as well as the improvement and performance of the product. Today, performance measurement has become a key feature in the development of successful software engineering applications, and productivity. [10]

Organizations generally choose software-based development process tools in order to more easily control comprehensive system developments. Agility is a need in a systematic service engineering as well as a co-design of requirements and architectural artefacts. [11]

Agile software development methodologies gaining the attention in the field of software engineering. Agile software development (ASD) has been in the software industry for years. Agile methodologies are gaining traction in the field of software development. There, it is coordinated to manage dynamic requirements, meet customer satisfaction, deliver products in a manner that meets customer requirements, and achieve the required quality. [12]

Scrum is one such modern agile software development framework that is widely used and known. Therefore, an important purpose of Scrum metrics is to help the software team and their managers to monitor the business development process, as well as business quality, productivity, predictability, the health of the product and the team. Scrum methodology consists of iterative and incremental sprint structures, and software development roles with the target determined prior to a sprint being started.

The Agile Methodologies and Scrum explained with more details on the section 3.2. Theoretical Methodologies.

To have a better idea on the problem, firstly we need to know what an interface is and what is a class so that we can compare WebDriver() Interface with RemoteWebDriver() class.

A class is a group of similar object types that share the same attributes and behavior. It also defines the abstract properties of an object, including properties (its attributes, fields, or properties) and behaviors (what the object can do, or methods, operations, or functions).

Like a class, an interface can have methods and variables, but the methods declared in interface are by default abstract (only method signature, no body).

Class	Interface
The ‘class’ keyword is used to create a class.	The ‘interface’ keyword is used to create an interface.
An object of a class can be created.	An object of an interface cannot be created.
Class doesn’t support multiple inheritance.	Interface supports multiple inheritance.
A class can inherit another class.	An Interface cannot inherit a class.
A class can be inherited by another class using the keyword ‘extends’.	An Interface can be inherited by a class using the keyword ‘implements’ and it can be inherited by another interface using the keyword ‘extends’.
A class can contain constructors.	An Interface cannot contain constructors.
It cannot contain abstract methods.	It consists of abstract methods only.

Table 1 Class vs Interface [13]

So, the class is used for object creation, encapsulation for fields, methods and interface is used to create a structure for an entity. Depending on the application both classes and interfaces has advantages and disadvantages. In our case the advantage of RemoteWebDriver() is running the browser no matter where the tests are running gives us a big advantage. Because in this way we can run our tests on local machines even though we do not have that browser.

WebDriver(): Selenium WebDriver() is a tool that allows you to automatically create and operate some test steps of websites through your web browser. If you want to test the correct and incorrect operation scenarios of the user registration page of your website, you can automate steps such as opening the user registration page, filling the input fields on the page, and clicking the button, with Selenium. In other words, it makes it easier for you to perform functional tests of your application. In this way, by saving different scenarios, you can have Selenium do these scenarios later instead of manually doing them yourself, and by examining the test results, you can say "results are successful" or "some scenarios have problems". [14] The biggest purpose of the development of WebDriver() is to help you maintain your test cases by seeing the changes made during the change of a certain part of the content of dynamic websites without reloading the entire page. The communication between clients and the server happens through a JSON wire protocol. For instance: when a command is given to open a browser with a specific URL, all the necessary information like the browser type, browser version, and the desired capabilities will be used to

create a JSON payload. The client will send this payload containing all the required information through JSON wire protocol over the HTTP client. The server would then identify the type of browser in which the command has to be executed and run the specified command on that browser.

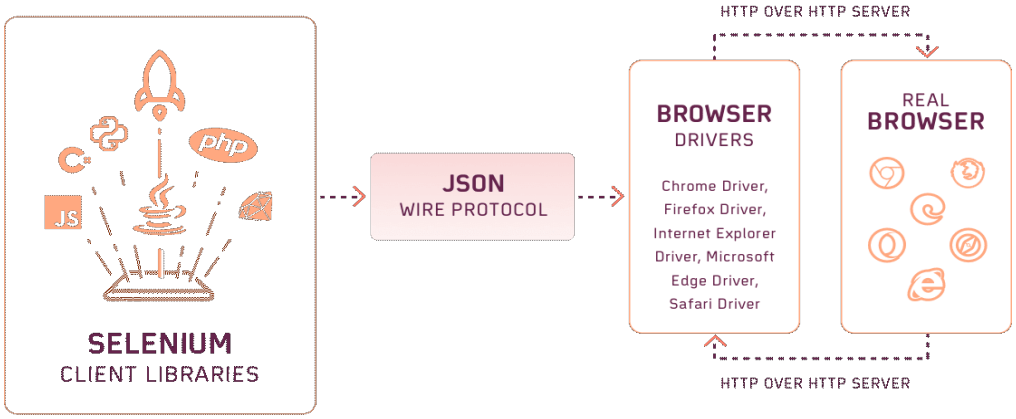


Figure 1 Selenium WebDriver() Architecture

RemoteWebDriver(): Selenium RemoteWebDriver() is used to execute the browser automation suite on a remote machine. In other words, RemoteWebDriver() is a class that implements the WebDriver() interface on the remote server. The browser driver classes like FirefoxDriver, ChromeDriver, InternetExplorerDriver, etc. extend the RemoteWebDriver() class. RemoteWebDriver() is the class that makes our tests run on remote machines. This means, you can run the test code on your machine, but the code execution will happen on another machine. [15]

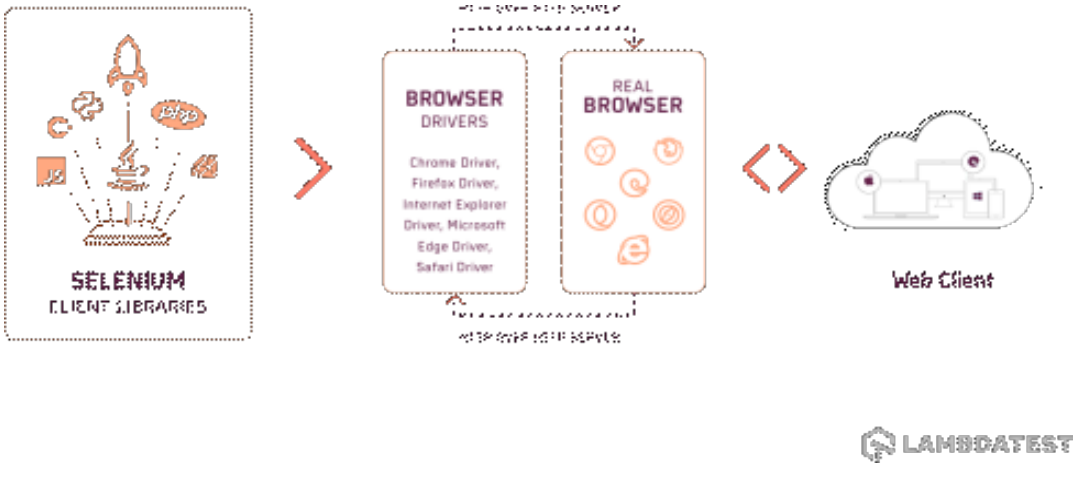


Figure 2 Selenium RemoteWebDriver() Architecture

The Difference Between Selenium Webdriver() and Selenium RemoteWebdriver():

Selenium RemoteWebdriver()	Selenium WebDriver()
A class that implements a Webdriver() interface.	An interface and to use this, org.openqa.selenium.* package is required.
It has some extra methods which are used for the implementation of the class.	It has a lesser number of methods as compared to Remote WebDriver(), used for the interface implementation.
It implements a WebDriver() interface	WebDriver()object is a browser
In this, the object is used to manage the browser which is present in the grid.	In this, many browsers are managed by the WebDriver()object.
Controls the desired browser in the node PC over Grid	WebDriver()object controls the web browser
RemoteWebDriver() object instantiation controls any browser in the node PC, which is on Windows, Mac, Linux, etc. platforms.	On instantiating the driver object, class implementations will be launched
Contains additional methods for class implementation	Comprises fewer methods for implementing interfaces
To test this, the remote machine is required.	It can be tested on the local machine.

Table 2 WebDriver() vs RemoteWebDriver()

When test cases are executed, the WebDriver() client libraries link with the browser drivers directly. On the other hand, if one tries to execute tests remotely, the WebDriver() client libraries communicate with the RemoteWebDriver() server. Then, the server links to either of the browser drivers the WebDriver() client requests. As in our test cases, the WebDriver() is created based on RemoteWebDriver() to use the RemoteWebDriver() when it is necessary but on the test classes almost never it is used.

The differences of RemoteWebDriver() from WebDriver(), RemoteWebDriver() has additional methods, and it can control the desired browser. My hypothesis is that since RemoteWebDriver() separates where the tests are running from where the browser is and allows tests to be run with browsers that are not available on the current OS, it could be much more efficient and faster to run the tests.

I tested this hypothesis during this thesis and explained the results on Results section but before explaining that I would like to explain core Methodologies of Software Development and Testing to make it easier to understand.

2.1. CPU Usage

To better understand what differences made by switching from WebDriver() to RemoteWebDriver() I also compared the CPU usage of each (see section Results).

The processor, also known as the central processing unit (CPU) is one of the most important components inside any computer. The processor serves as the brain of the operation. It sends instructions to all the other pieces of hardware in your device and is responsible for executing each task.

3. Methodologies

In this section I will define the technical and theoretical methodologies which I used and learned during the internship and writing of thesis.

Java EE: Java is an analytically typed object-oriented programming language, intended to be independent as much as from the underlying platform thanks to a virtual machine system. Java Enterprise Edition, or shorter JavaEE, is a wider version of the platform that involves many beneficial properties for web programming.

Also, Java EE is a structure to build enterprise software and it encases not only web content but also database, web services, queues, and so on. In Invoice Channel® it is used for both the frontend and the backend part. [16]

SQL: Acronym for Structured Query Language, it is a language that standard to be used to define and manage the data kept in a relational database. [17]

Spring: Spring is an open-source framework for applications to be developed on a Java platform that deals with reverse of control and dependency injection. [18]

Selenium: Selenium is an open-source and free testing tool used to test web applications on different platforms and browsers. Since it takes a lot of time to run the tests manually, automating it will save us time. Selenium can be used to run tests automatically in line with the given commands. [19]

Vaadin: Vaadin is a web framework for Rich Internet Applications (RIA) with a complex widget accessibility from the web browser. It also "hides" from the developer the need to be concerned about different browser behavior, mobile, tablet, and so on. It has server-side architecture, which means most of the work on servers. AJAX technology works on the browser side to ensure that user have a rich and interactive experience. On the client-side, Vaadin is built on top and can be extended with Google Web Toolkit. Thanks to the infrastructure it offers, it enables the creation of powerful, performance, and fast web software as if developing desktop applications. With Vaadin it allows getting rid of many details about JavaScript and HTML. But also, you can integrate many JavaScript libraries with Vaadin. [20]

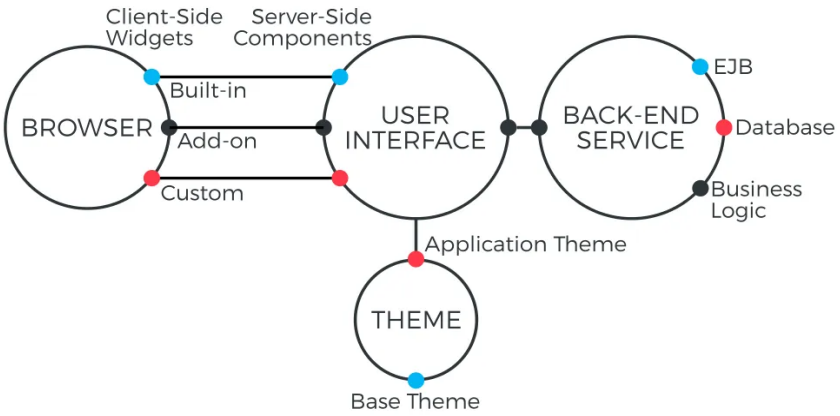


Figure 3 Vaadin Application Architecture

3.1.Development Tools

Eclipse: Eclipse is a multi-platform, multi-language integrated development environment. During the internship it was used for both unit testing and development. [21]

Tomcat: Apache Tomcat is an open-source implementation of the Java servlet, which provides a software platform for running web applications developed in that language. [22]

Docker: Docker is a technology that provides virtualization and, is a software platform that lets you quickly build, test, and deploy your applications. Docker takes the image of the last installed versions of the software and makes them usable again as containers. [23]

Maven: Maven is a dependency management tool for Java-based projects. It uses a construct called POM to describe the dependencies between the project and the various versions of the required libraries and automatically downloads libraries and plugins from a shared repository. It also manages the building of the project (compiling), the packaging (i.e.: WAR, EAR, JAR, and so on), can automatically run unit tests, and the deployment phase on an application server. [24]

Bitbucket: Bitbucket is a tool developed by Atlassian which includes: a hosting service for projects that use GIT for version control and a continuous delivery system. [25]

Jira: Jira is a tool developed by Atlassian for managing projects developed with agile methodologies. In carrying out the internship it was used to manage the tickets in charge. [26]

Bamboo: Bamboo is a Continuous Integration tool. It allows us to create automatic version management and continuous distribution models for software applications. It is written in Java language and the Server version runs on Tomcat. [27]

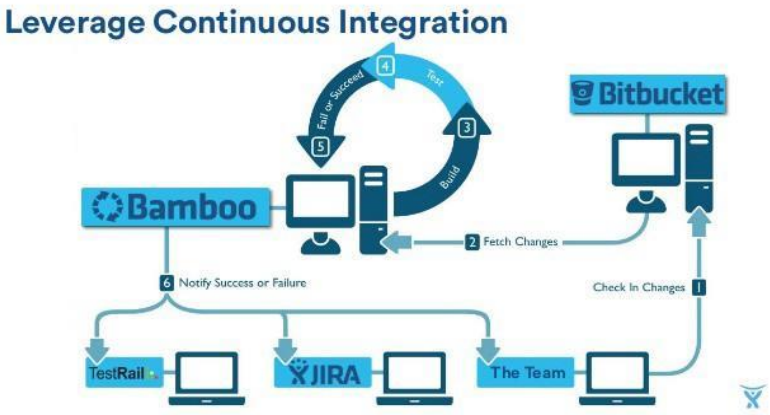


Figure 4 Leverage Continuous Integration

3.2.Theoretical Methodologies

Software Development Life Cycle: Software Development Life Cycle is a path that covers software from planning to development, testing, and end-of-life. [28] In this lifecycle model, at the very beginning of the software development process, starting from the installation of a source control repository where the team members will develop together, automatically sending the developed application to dev/test/pilot environments, version management, preparation of test reports, measuring code quality, performing code review operations, versioning of the data layer and running tests, environment-based configuration management, preparation of process management for production deployment and hotfix operations.

SDLC – Stages: We can generally classify the basic steps of the software life cycle as "Planning, Analysis, Design, Implementation, Maintenance".

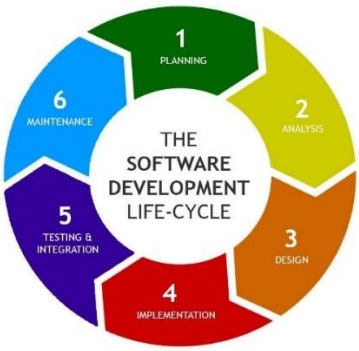


Figure 5 Software Development Life Cycle Stages

Benefits of SDLC: Properly done SDLC can allow for the highest level of management control and documentation. Developers understand what they need to build and why. All parties agree on the goal in advance and see a clear plan to achieve it. Everyone understands the costs and resources required. [29]

Continuous Integration / Continuous Development: Continuous Integration is the method used to control the system is in working condition after some changes made on the code and that changes causes no problem. Unit tests are used to uncover problems and breaks. With continuous integration, it is ensured that a working version is always created because of the work done on the code by the programmers. The changes made are part of a new structure so, errors in the tests mean that the change made breaks the system. In this case, all programmers are informed, and the error is eliminated as soon as possible.

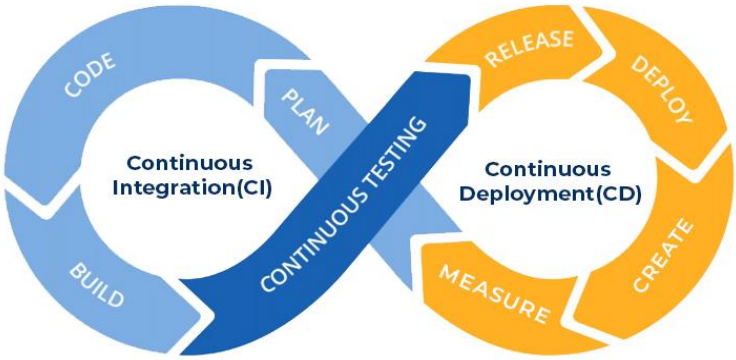


Figure 6 CI/CD Pipeline

Continuous delivery is a natural extension of continuous integration. Continuous delivery is the automatic way to throw a successful build into an environment. There is a slight difference between delivery and distribution. Delivery is done manually, while distribution is automatic. If continuous delivery is properly implemented, customers will have a standardized testing structure.

CI/CD Principles:

- Designing the system to support periodic releases
- Apply test-driven while developing code.
- Test your system often.
- Always have a separated team for the quality and steady progress of the software.

Test-Driven Development: Test-Driven Development (TDD) is a technique that argues that after determining which codes will be written for what purpose within the project, first the tests and then the code should be written accordingly. The advantages of the TDD approach are faster feedback, higher acceptance rate, lower project scope and over-engineering, customer-centric and iterative processes, and modular, flexible, and maintainable code. [30]

TDD approach consists of 5 stages:

- Think and create a test case.
- Make sure the programmer test fails before developing the unit.
- Build the unit and pass the programmer test.
- Ensure all programmer tests pass, along with prebuilt ones.
- Scan all units with your latest development and refactor them at necessary points. [30]

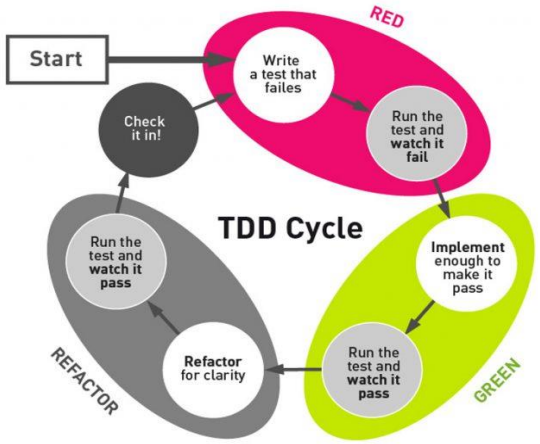


Figure 7 Test-Driven Development Life Cycle

Agile: The first foundations of this method are based on the software development processes at IBM in 1957 by William Royce. The Agile Method was introduced by E.A. Edmonds in an article titled "Software Development for Compatible Systems" in 1974. In 2001, 17 people known in the software world came together and after 2 days of work, they published the "Agile Software Principles" together with the "Agile Software Development Manifesto". [31]

The Agile method is a method that divides the main project into subprojects, which ensures that we always have minimum-featured software running and is a special approach to project management used in software development. This method helps teams respond to the

unpredictability of software development processes. It uses incremental, iterative work sequences, commonly known as sprints.

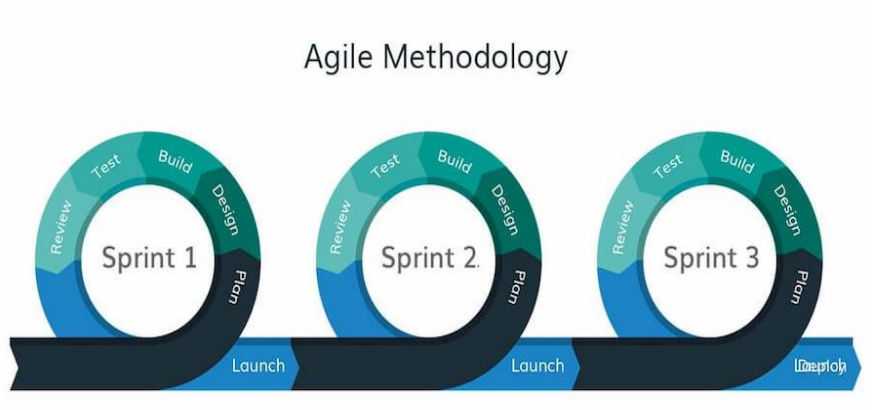


Figure 8 Agile Methodology

The founding principles of this methodology are described in the Agile Manifesto and can be summarized in four key points:

- individuals and interactions: organization and autonomous motivation are important, as are personal interactions, for example, sharing the same place of development.
- working software: a product that works is more useful and better accepted than paper documents presented to buyers during meetings.
- collaboration with buyers: interaction with customers is extremely important as the requirements cannot be fully identified at the beginning of the software development cycle.
- responsiveness to change requirements may vary during construction so it is essential to provide quick responses to change and continuous development.

The Agile Project Management manifesto and principles are based on quick and easy responses to changing conditions. In this context, each of the Agile Project Management methodologies has a cycle that repeats at regular intervals (minimum once every two weeks). This model does not allow changes during the project. Scope, time, and resources are determined from the very beginning of the project; While the project is in progress, it is not left out of the determination. Since the project boundaries are determined strictly, it ensures that the project is completed quickly without any distraction. The tasks of the team involved in the project are specific and everyone concentrates only on their own task. Thus, it ensures the stable progress of

the project. The project is delivered to the customer on the specified date. Agile is used to manage complex software processes and it breaks the whole; follows an iterative method. It enables you to reach the target with regular feedback and planning. Communication and teamwork are very important.

Scrum is a progressive framework to predictability and controlling risk. It is founded on 3 basic principles below:

- Transparency: The progress of the project, problems, and developments should be visible to everyone.
- Check: The progress of the project is checked regularly.
- Adaptation: The project should be able to adapt to the changes that can be made. [32]

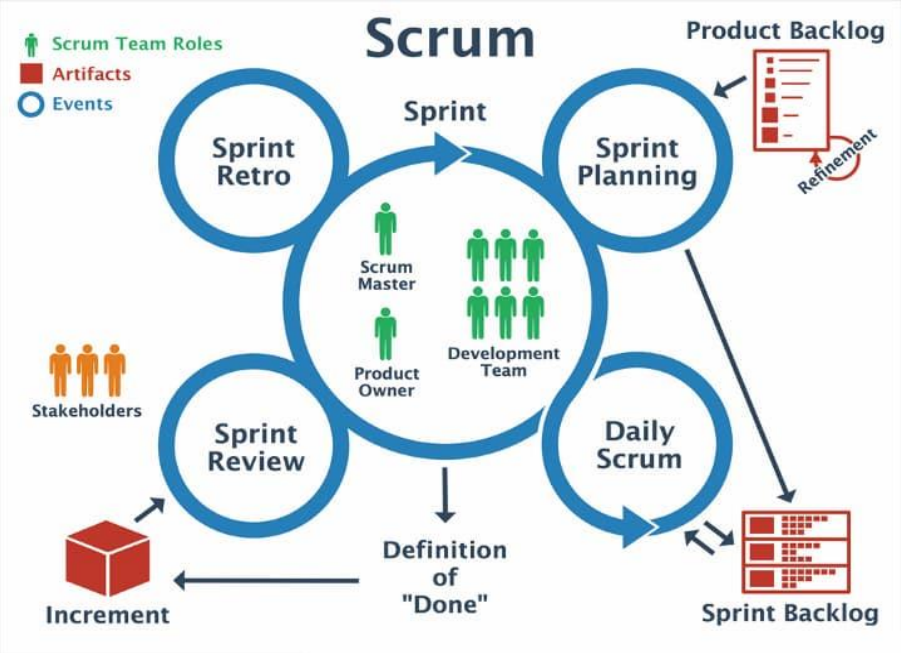


Figure 9 Scrum Life Cycle

Scrum Team: It consists of the Product Owner, Development Team, and Scrum Master. The team organizes itself. Thus, teams that are in harmony within themselves get more successful results. Scrum's team model is designed to optimize flexibility, creativity, and efficiency. Monthly meetings are required for work in the sprint backlog. These jobs are expected to be done in one sprint. So even though we call it monthly meetings, Sprints can be 1-2 weeks and at the end of each one there's a retrospective review. Daily standing Scrum meetings are held in which each one

explains the work done the previous day, the work they intend to do during the day and any obstacles and difficulties encountered. The latter, if necessary, will be studied separately to make sure it goes on schedule. These meetings usually last 15 minutes.

4. Test

Testing is the process of experimenting or evaluating a system, either manually or automatically, to verify that it meets specified requirements or to identify differences between expected and observed results. Software testing, on the other hand, covers the dynamically performed verification activities to meet the expected behavior of software from an infinite number of work areas, with a limited number of appropriately selected tests. [33]

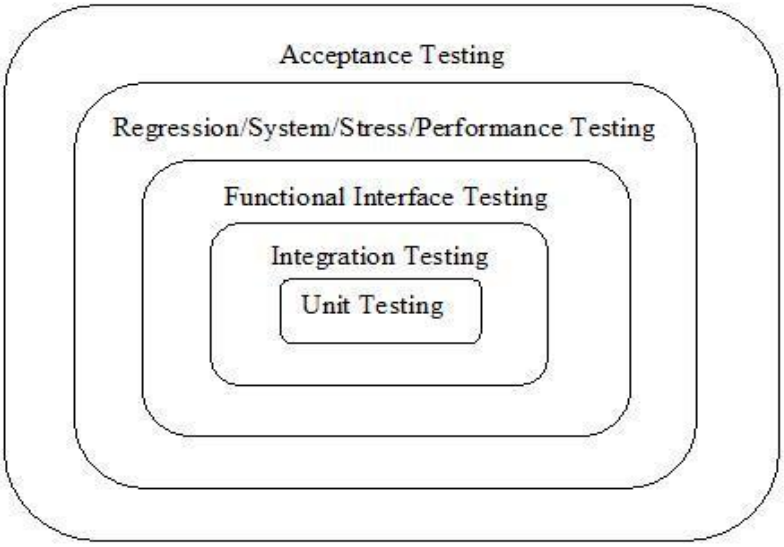


Figure 10 Types of Tests and Hierarchy of Tests

4.1. Software Testing Process

In general, software projects are developed by the following: analysis => design => coding => testing => product processes. Although all processes follow each other in this way, the testing process never waits for the coding process to end. An ideal software testing process should be analysis => design => test preparation process => coding => dynamic testing process => test termination => product.

There are two main classifications of software testing: functional and non-functional.

Functional testing is about testing how a product works. Performing a functional test entail checking and testing every single function of the software to ensure you get the expected results.

Non-functional testing is about testing non-functional aspects of software for usability, performance, security, reliability, and more. This test is performed after functional testing. Non-

functional testing does not focus on whether the software works, it focuses on how well it works. [34]

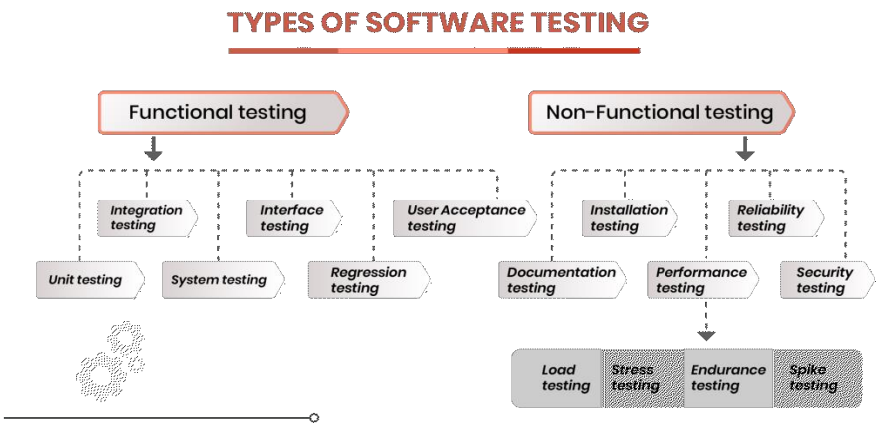


Figure 11 Types of Software Testing

Functional Testing is a type of test that verifies whether each function of the application we are testing is working in accordance with the given requirements. It checks the results of a given set of data by comparing them to a set of results. Functional tests are not concerned with intermediate results or side effects. They are only result oriented.

Unit Testing is the first stage of the dynamic testing process; it also constitutes the most important stage of this process in terms of early detection and correction of errors. In this micro-scale test, special functions, or code modules (functions, data structures, objects, etc.) are tested. This testing is done by programmers, not testers, and it is necessary to know the details of the program code and its internal design. It is quite a difficult test if the application code is not in a very well-designed architecture.

Integration Testing is to test whether different components of an application work together in harmony. They can be in the form of components, modules, standalone applications, and client/server applications. This type of testing is especially applied in testing client/server applications and distributed systems.

In addition, the process of constantly testing the application as new functional elements or program modules is called "Incremental Complementary Testing". Tests performed by testers and/or programmers.

Regression Testing is used to control the changes made to the code running live. These changes may be a new function, bug fix, or performance improvement. Regression tests are usually

performed when changes are in their final stages and before the new version of the software is released. The primary purpose of regression tests is to check those critical areas of the application still work as expected. [35]

Non-Functional Tests are a type of test in which non-functional features of the application are tested. The aim is to determine whether the system is ready or not. The quality characteristics of the components or the system are tested. It is as important as functional testing in the quality and correct operation of the software. For example, how many users can use the system at the same time, is the system secure enough, and whether the system is tested to answer such questions. [36]

Performance Testing determines the system's expectation of the best possible (optimal) performance under a given infrastructure configuration. The purpose of performance testing is not to find faults, but to eliminate performance bottlenecks and measure the quality characteristics of the system. Performance testing mainly focuses on the following specific features of a software. These; speed, stability, and scalability. [37]

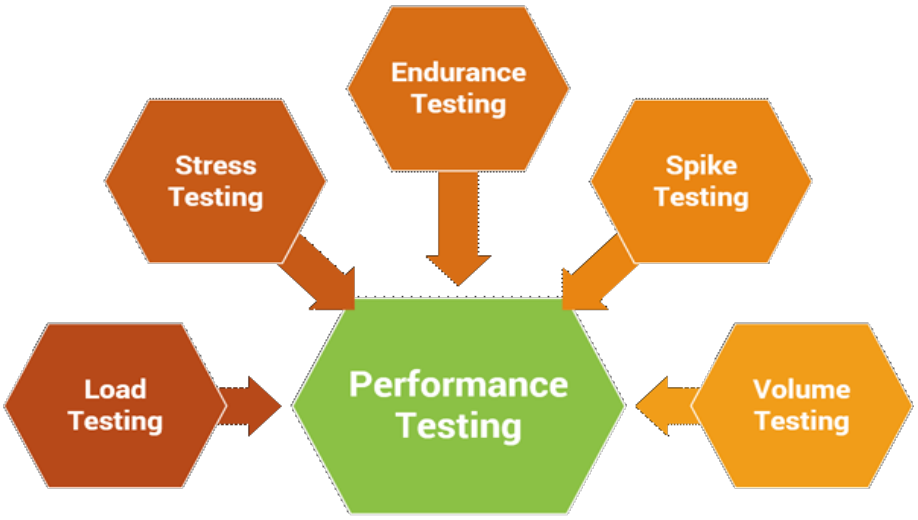


Figure 12 Types of Performance Testing

Test Pyramid is the road to a successful software project is through a successful testing process. One of the best ways to do this is the Test Pyramid. The Test Pyramid refers to the different test tiers related to the classification of tests in terms of scope and size.

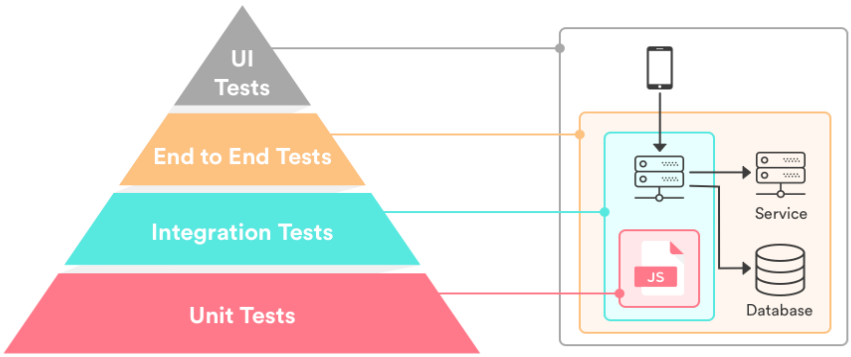


Figure 13 Test Pyramid

The lower portion of the pyramid is more automated, making those tests faster. The top of the pyramid is for slower and more manual test scenarios. The higher you go on the pyramid, the fewer the tests.

Mike Cohn, who came up with the concept, made the classification as unit, service, and interface. Unit tests are the units of code that make up the service, tests written for its parts. Service tests are API-level tests. Interface tests are tests written on the user interface. [38]

The key to the test pyramid is to write lots of small and fast unit tests and write some more coarse-grained tests and very few high-level tests that test your application from end to end.

4.2. Test Performance

When we say performance, we mean more than one thing, so the word performance is an umbrella name for many different quality criteria clustered around “speed”. Test Automation is an important step of Continuous Deployment. Automation reduces bug fix costs by testing early and often testing principles. Automation projects also increase quality and reliability. But test execution duration problem comes up with the benefits of the automated test. So, for performance improvement we should look at the below criteria:

- End-user response time: It is the system's response time for the actions of the end-user while using the system. It is the factor that interests the end-user the most and bothers him when it is long and disrupts the rhythm of work. This is what is meant most of the time when "the system is slow", but it can have very different reasons. The primary objective of performance improvement studies is to reduce the end-user response time to reasonable levels and to provide acceptable end-user performance on average. [39]

- Scalability: It refers to how the behavior of the application changes under increasing load. The two main factors affecting scalability are the growing number of live users and the growing amount of data. Systems with poor scalability also have a poor end-user response time. But this situation does not always occur, sometimes it becomes more visible at certain times. This is entirely since the system load is constantly changing. In performance improvement efforts, scalability is considered a second priority, keeping end-user response time within reasonable limits. [40]
- Memory footprint: It is the memory usage of the application. It usually occurs as a memory leak and is the factor that most affects the scalability of enterprise applications. As memory leaks cause the garbage collection mechanism to be activated frequently and for long periods of time in Java-developed applications, and this activity consumes serious resources, it ultimately affects the end-user response time negatively. If memory usage is not improved, it will eventually eliminate scalability altogether and make systems need to be rebooted frequently. It is difficult to achieve healthy scalability in performance improvement studies without addressing memory usage or optimizing garbage collection activity. [41]
- Throughput: The amount of work produced by the application in each time. Especially for mass printing, invoicing, image processing, data transfer, etc. It becomes much more important with jobs because it's how many of those jobs are done in each time, rather than printing a report or generating a subscriber's invoice. Therefore, often this criterion is of a conflicting nature with the end-user response time. For example, a system that generates a subscriber's invoice in 10 seconds may not satisfy the end-user with this performance. But the same system produces 1,000,000 invoices per hour in batches, performing parallel processing in the back, and this can be a perfectly acceptable throughput. Studies to increase the work produced in performance improvement studies may be required depending on the nature of the application. [42]
- Startup time: This is the time from when the application is first started until it is ready to respond to user requests. This criterion is most important for client-side applications. For example, as end-users, we are increasingly paying attention to how long it takes for a word processor on your personal machine to be ready for use after clicking its icon. Therefore, for example, we replace the mechanical hard disks that we have been using for years, with

SSDs that are much faster. However, the fact that the applications on the server (server) side stand up in 15-20 minutes, for example, may not always be seen as a problem. For this reason, this criterion is generally not considered a priority in performance improvement studies. [43]

We can follow the steps below to make improvements in a software system with poor performance:

1. Identifying and listing performance problems
2. Measuring the performance of each problem and determining the improvement target,
3. prioritizing problems,
4. Addressing the highest priority problem
5. Identifying and working on the points of the problem that require improvement,
6. Measuring improved performance
7. Repeating the 5th and 6th items until the performance target is met,
8. Continuing from point 4.
9. With such a process, we can determine where we have come from, as well as increase the possibility of making value-added improvements.

To achieve target performance criteria using the process described above, three different solutions are used:

- Code change: It is to replace the code snippets that are problematic in terms of performance with more efficient ones. Finding a solution with changes to the code is often the fastest found and the fastest improvement. Such code changes can be both algorithmic and language based. But the impact of such changes is often minor.
- Architectural change: It is the organizational changes to be made in the sub-parts of the software system. Common code changes often translate into architectural change. Architectural changes are both costly and provide a much better improvement rate. If a system that does not use cache starts to use the cache, or for example, setting up partitions in the database, etc. software architecture changes are among the most frequent improvement studies. Architectural changes do not always occur as code changes, such as improving hardware infrastructure to have faster CPUs or high RAM, arranging network infrastructure, or strengthening clients are such architectural improvements.

- JVM tuning: Observing the behavior of the JVM, which is the runtime mechanism of Java, and arranging it to meet the needs of the system. JVM tuning work, in general, is the regulation of the parameters that the JVM receives while standing up. These parameters are memory, garbage collector, JIT, etc. It is about topics and usually comes with serious innovations with each new Java SE release. For this purpose, the JVM(s) are observed at runtime, most of the time the parameters are brought to the optimum point with a set-observe-change cycle. JVM tuning should be done especially in the real environment, before going live, and it should be repeated at certain intervals. [44]

5. Project

5.1. Description of Project

During my internship my supervisor, and my team leader expected me to make some improvements on tests, try to find some ways to make the tests faster, or find what might be making the test slower and how to discard these parts.

To do that firstly I examine the test runs to see which tests are slow. And I made a list which you can see below:

Test Case Name	Test Name	Average Duration
006 – GO LIVE	ProfiloOperazione	8 minutes
008–CP – CONFIGURAZIONE	SPCicloPassivo	56 minutes
008–CP – CONFIGURAZIONE	CapAdminCP	8 minutes
010 – CP – RICEZIONE FTP	AccettaFattura	10 minutes
011 – CP – RICEZIONE WS	AccettaFatturaWS	8 minutes
012 – WEB SERVICE	TestWebService	11 minutes
013 - WEB SERVICE 2	TestPushSynchronousRemoteItem	8 minutes
013 - WEB SERVICE 2	TestSetSoggettoProduttore	4 minutes
014 – 2000 ORGANIZATION	InserisciSP	74 minutes
018 – INTEGRAZIONE CON WS LA	CronConservazioneWS	17 minutes
019–WIZARD ANAGRAFICHE	WizardAnagrafiche	16 minutes
021–CP–TEST CONSERVAZİONE	ConfConservazioneCP	8 minutes

Table 3 Slowest Test on Invoice Channel Tests

Accordingly, to the *Table 3*, some tests are taking enormous amount of time and as the test team leader informed me some of those tests are not finished and for now normal that they are taking this much of time.

5.2. Task

My task for this project was to improve the test performance, because Invoice Channel® is a big project and involves too many test cases and it is very important to run this test as fast and as affective we can. So firstly, I made a list of the tests running for more than 2 minutes in the

specific part of the project Invoice Channel® which you can see the list on *Table 1*. To see the execution times of the test I used Bamboo results. Every night the whole Invoice Channel® tests are ran by automatically on Bamboo, so I checked last 10-night runs and listed all the testes that are taking more than 2 minutes. Out of 121 test cases, 12 tests were almost taking 10 minutes to run. After that, I analyzed why might these tests take so long to run and what's the common thing for them. I made a document which the document was containing lists for every class that might be causing the slowness.

With all the analyzes I made I have shared the document with my team leader. And after discussing with him, we have decided to work on a specific test class because that test class was involving both WebDriver() and Thread.Sleep() which are the most common things on the test that might be making the tests slower.

The test case that we decided to work on is called "AccettaFattura" which essentially it enters as admin. It enters the details of the passive invoice and reads the producer. It logs out and logs in with the User_ALIASsp user as the admin user which is have the possibility to approve the invoice received. I approve the invoice and shows a text that it can't be done again. The support files are copied from a bamboo folder to the \tifnvl6ic1\ic\ic\ftp\DatiDaSdITest folder. The .enc extension is also added to the copied file so that the crontab can act on the file. A media can contain multiple invoices. Invoices are identified by the invoice number. These are taken from the CSV file mediaNfatt.csv which contains the relationship between the media file name and its invoice numbers. The AccettaFattura class contains two methods:

One is called *"a00VerificaPresenzaSchedulazione()"* which verifies if the crontab (scheduling of processes) is present; without this crontab, the invoice will never be loaded. Specifically:

- Opens a Chrome browser and logs in as administrator
- Navigates to the process menu & opens the form
- Searches for a scheduled process with the command SDI_FOLDER_SCAN (scan a folder on the FS and fetch invoices from there).
- If no such process exits with failure
- Search if this process is marked as active, if not mark it as active
- If there is no crash so far, the test is considered a success.

The other is called *"a01CicloPassivo()"* which in essence waits for some invoice to be received which received means importing the invoice from the file to invoicechannel, put in the database, and associated to an SP.

- Initially simulated what a user will do outside Invoice Channel®: move some "invoice files" into the folder "scanned" by the process described in the previous method

- Subsequently waits for the process to "pass by grab the invoice from the folder and import it into IC" For the system to scan the file the tests wait 1,5 minutes with Thread.Sleep(90000).

- After that, it opens IC in the browser

- Goes to "ciclo passivo invoice form" because the test is for the passive invoices.

- Searches the invoice-by-invoice number "Fattura Numero"

- If there is not an invoice with that number then the test fails, if there are goes on to the next step

- if I got here, it means the invoice is imported so as the next step it needs an operation to finalize the import (protocollazione)

- Presses to the "protocolla button"

- It logs out and logs in back with the user associated with that invoice (this is a user that can "approve" the invoice)

- Process to the to approve the invoice (menu ricerche ....)

- If the user sees the newly accepted invoice, then it is a success, if not then the test has failed.

To improve the performance of the test I decided to chance WebDriver() with RemoteWebDriver() as the reasons explained on the section 4.2.1.

After seeing the results of the AccettaFattura, we have decided to apply RemoteWebDriver() to another class which is not slow but also can be faster. The class is called "CreaSoggettiProduttori" which with root user it reads the csv soggettoProduttore and creates all the subject producers present in the file. In this class we have only one method called a00TestCreaSoggettiProduttori().

Primarily, the a00TestCreaSoggettiProduttori() method first opens the login page and logs in as root user. Then in order, from the menu it clicks to 'Amministrazione', opens the 'Soggetto Produttore' and starts creating new subject producers with the information taken from csv file.

5.3. WebDriver() vs RemoteWebDriver()

As can be seen in Figure 14, WebDriver() is an interface and RemoteWebDriver() is a class that implements WebDriver() interface.

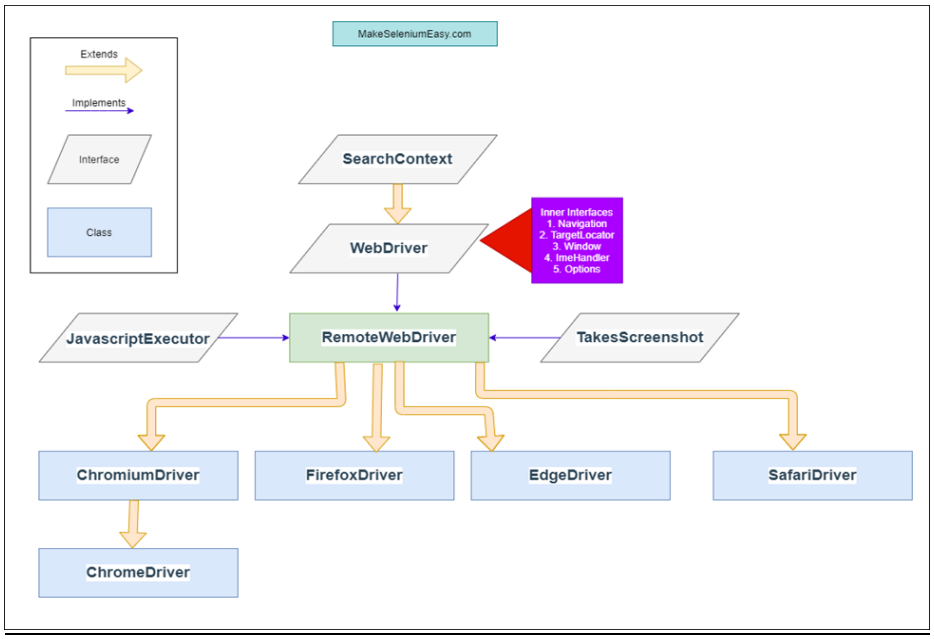


Figure 14 WebDriver() Hierarchy Diagram

Selenium WebDriver() allows controlling web browsers using different language bindings such as Java, JavaScript, Python, C#, or Ruby. In end-to-end tests, the Selenium WebDriver() Application Programming Interface (API) calls are typically embedded in test cases using a unit testing framework. [45]

Selenium WebDriver() supports multiple web browsers and support for Ajax applications. The main goal of the selenium WebDriver() is to improve support for modern web application testing problems. [46]

The RemoteWebDriver() class implements the WebDriver interface to execute test scripts through the RemoteWebDriver() server on a remote machine.

When test cases are executed, the WebDriver() client libraries link with the browser drivers directly. On the other hand, if one tries to execute tests remotely, the WebDriver()client libraries communicate with the RemoteWebDriver() server. Then, the server links to either of the browser drivers the WebDriver() client requests. As in our test cases, the WebDriver() is created based on RemoteWebDriver() to use the RemoteWebDriver() when it is necessary but on the test classes almost never it is used.

RemoteWebDriver() class contains additional methods that are not declared by the interface WebDriver(). Such as “getSessionId()”, executeAsyncScript(java.lang.String script, java.lang.Object... args) which executes an asynchronous piece of JavaScript in the context of the currently selected frame or window; removeVirtualAuthenticator(VirtualAuthenticator authenticator) which Removes a previously added virtual authenticator.

RemoteWebDriver() Code and Explanation:

To create the RemoteWebDriver() we need to set some capabilities and to do that I used the similar capabilities that when the first time how the project build WebDriver() is using but this time I created a RemoteWebDriver(). I created a new method called “RemoteWebDriver() createDriver()” on BaseChrome() class which returns a RemoteWebDriver() when it is called.

BaseChrome() class is to control the Chrome operations like setUp() to set the capabilities and details about driver, openIC() to open the login page of Invoice Channel, and verificaPopupErrore(..) to verify the popup errors. To the BaseChrome() class I added two methods to reach the RemoteWebDriver() from every class that extends the BaseChrome() class. These two methods are “openICRWD(RemoteWebDriver() driver)” which calls the RemoteWebDriver() driver that we should create at the other method as I created “createDriver()”.

In createDriver() I set all the elements that are needed to create the RemoteWebDriver(). To do so I first put all the requirements into a ‘HashMap<String, Object>’ and then I created an instance of ChromeOptions, then I can use the 'setCapability()' method for setting ChromeDriver-specific capabilities and pass the ChromeOptions object into the RemoteWebDriver() constructor. Lastly, I used DesiredCapabilities Class to set the capabilities for RemoteWebDriver(); which DesiredCapabilities Class is used to define basic test requirements such as operating systems, browser combinations, browser versions, etc. within Selenium test scripts.

```
DesiredCapabilities capability = new DesiredCapabilities();

String downloadDirectory = "\\\\" + System.getProperty("server.download.upload.licenza")
        + "\\ic\\ic\\download";

capability.setBrowserName("chrome");
HashMap<String, Object> chromePrefs = new HashMap<>();
chromePrefs.put("profile.default_content_settings.popups", 0);
chromePrefs.put("download_restrictions", 0);
chromePrefs.put("download.default_directory", downloadDirectory);
chromePrefs.put("download.prompt_for_download", false);
chromePrefs.put("safebrowsing.enabled", true);
chromePrefs.put("credentials_enable_service", false);
chromePrefs.put("profile.password_manager_enabled", false);
ChromeOptions options = new ChromeOptions();
options.setExperimentalOption("prefs", chromePrefs);
options.addArguments(Arrays.asList("--disable-infobars", "--test-name=IC", "--test-type=browser",
        "--disable-default-apps", "--disable-popup-blocking", "--js-flags=--expose-gc", "--start-maximized",
        "--no-sandbox", "--ignore-certificate-errors", "--homepage=about:blank", "--no-first-run",
        "--incognito", "--lang=it-IT", "--safebrowsing-disable-download-protection",
        "--safebrowsing-disable-extension-blacklist", "--disable-permission-action-reporting",
        "--disable-web-security", "--allow-running-insecure-content", "force-device-scale-factor=1",
        "high-dpi-support=1"));
capability.setCapability(ChromeOptions.CAPABILITY, options);
capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
RemoteWebDriver driver = new RemoteWebDriver(testingbotdotcom, capability);
```

Figure 15 Code of WebDriver() Capabilities

To call the RemoteWebDriver() to open a Chrome from every class that we wanted to use, I also added a method called “openICRWD(RemoteWebDriver driver)” which calls the driver from “RemoteWebDriver createDriver()”.

```
protected RemoteWebDriver createDriver() throws Exception {
    try {
        URL testingbotdotcom = new URL("http://" + clientKey + ":" + clientSecret + System.getProperty("hub.url"));
        Log.info("[URL]" + testingbotdotcom.toString());
        baseUrl = System.getProperty("url.application");
        DesiredCapabilities capability = new DesiredCapabilities();

        String downloadDirectory = "\\\\" + System.getProperty("server.download.upload.licenza")
                + "\\ic\\ic\\download";

        capability.setBrowserName("chrome");
        HashMap<String, Object> chromePrefs = new HashMap<>();
        chromePrefs.put("profile.default_content_settings.popups", 0);
        chromePrefs.put("download_restrictions", 0);
        chromePrefs.put("download.default_directory", downloadDirectory);
        chromePrefs.put("download.prompt_for_download", false);
        chromePrefs.put("safebrowsing.enabled", true);
        chromePrefs.put("credentials_enable_service", false);
        chromePrefs.put("profile.password_manager_enabled", false);
        ChromeOptions options = new ChromeOptions();
        options.setExperimentalOption("prefs", chromePrefs);
        options.addArguments(Arrays.asList("--disable-infobars", "--test-name=IC", "--test-type=browser",
                "--disable-default-apps", "--disable-popup-blocking", "--js-flags=--expose-gc", "--start-maximized",
                "--no-sandbox", "--ignore-certificate-errors", "--homepage=about:blank", "--no-first-run",
                "--incognito", "--lang=it-IT", "--safebrowsing-disable-download-protection",
                "--safebrowsing-disable-extension-blacklist", "--disable-permission-action-reporting",
                "--disable-web-security", "--allow-running-insecure-content", "force-device-scale-factor=1",
                "high-dpi-support=1"));
        capability.setCapability(ChromeOptions.CAPABILITY, options);
        capability.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
        RemoteWebDriver driver = new RemoteWebDriver(capability);

        String usernameProperty = System.getProperty("userName");
        if (usernameProperty != null && !usernameProperty.isEmpty() && !usernameProperty.contains("$")) {
            userName = usernameProperty;
            if (!"root".equals(usernameProperty)) {
                password = "!2eA56789";
            }
        }
        setUpScreenshotParams();
        return driver;

    } catch (Exception e) {
        Log.error("Errore nnela fase di setup - Hub / nodo probabilmente spenti", e);
        throw e;
    }
}

protected void openIC() {
    getDriver().manage().timeouts().implicitlyWait(DEFAULT_IMPLICIT_WAIT, TimeUnit.SECONDS);
    getDriver().manage().window().maximize();
    // getDriver().manage().window().fullscreen();
    getDriver().get(baseUrl);

}

protected void openICRWD(RemoteWebDriver driver) {
    driver.manage().timeouts().implicitlyWait(DEFAULT_IMPLICIT_WAIT, TimeUnit.SECONDS);
    driver.manage().window().maximize();
    // getDriver().manage().window().fullscreen();
    driver.get(baseUrl);

}
```

Figure 16 Code of createDriver(), openIC() and openICRWD()

Accordingly, to the code on Figure 16 above, I can call RemoteWebDriver() from any other class that extends BaseChrome() and apply the methods.

6. Results

To show the results of this task I used the open-source JDK Mission Control (JMC) tool which is a production-time profiling and diagnostics tool. [47] It includes tools to monitor and manage your Java application with a very small performance overhead. JMC consists of the following client applications and plug-ins:

- JVM Browser shows running Java applications and their JVMs.
- JMX Console is a mechanism for monitoring and managing JVMs. It connects to a running JVM, collects, displays its characteristics in real-time, and enables you to change some of its runtime properties through Managed Beans (MBeans). You can also create rules that trigger certain events (for example, send an e-mail if the CPU usage by the application reaches 90 percent).
- Flight Recorder (JFR) is a tool for collecting diagnostic and profiling data about a running Java application. It is integrated into the JVM and causes a very small performance overhead, so it can be used in production environments. JFR continuously saves large amounts of data about the running applications. This profiling information includes thread samples, lock profiles, and garbage collection details. JFR presents diagnostic information in logically grouped tables and charts. It enables you to select the range of time and level of detail necessary to focus on the problem. Data collected by JFR can be essential when contacting Oracle support to help diagnose issues with your Java application. [47]

6.1. Performance Test and CPU Usage

The better the CPU, the more tasks you can simultaneously perform without a hitch so, I measure the performance of my test using Flight Recorder. With Flight Recorder I recorded CPU Usage while running the same test class with WebDriver() and then with RemoteWebDriver(). You can see the result of the performance test as a graphic:

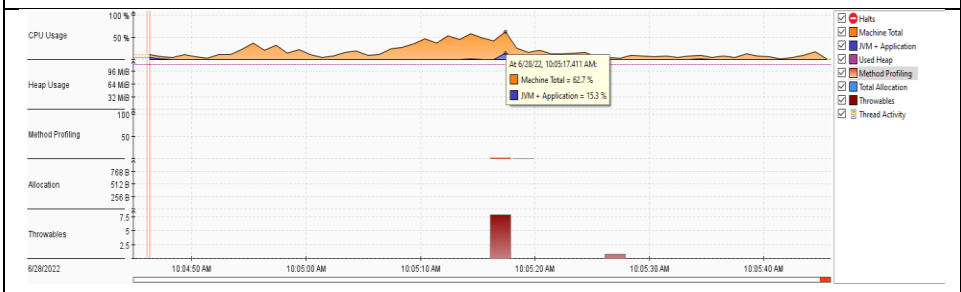
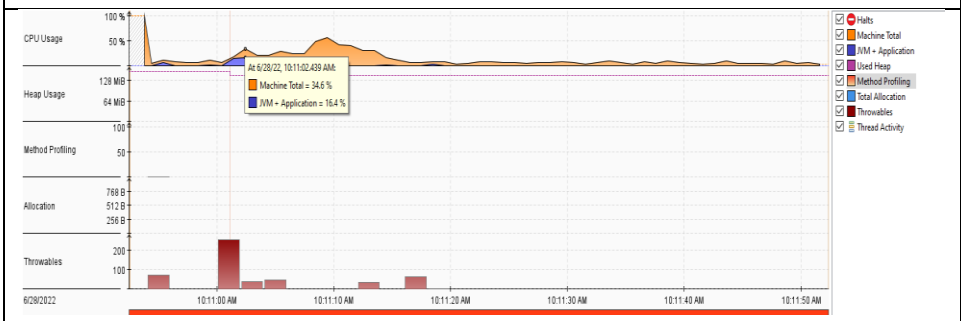
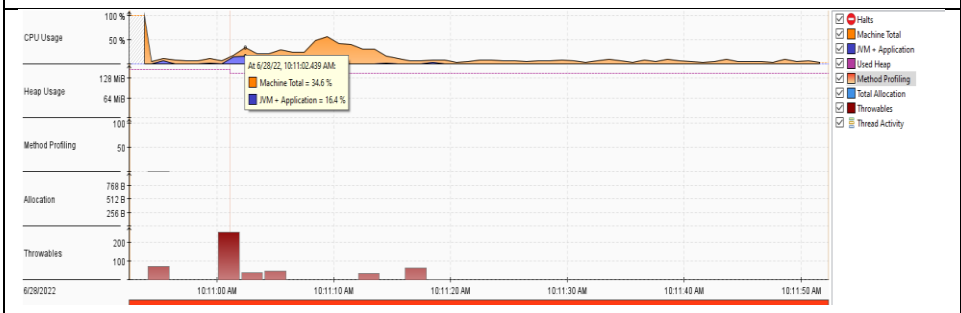
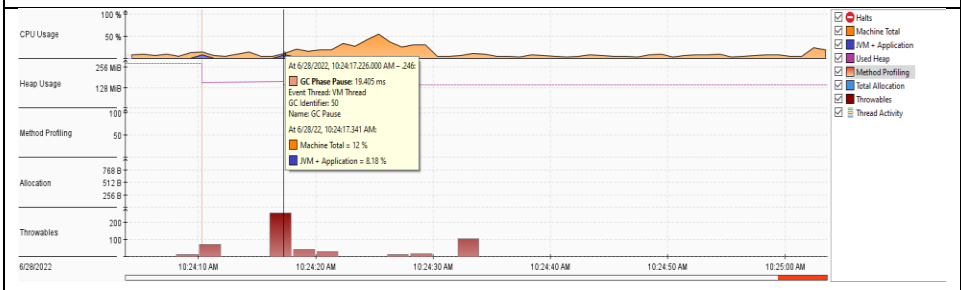
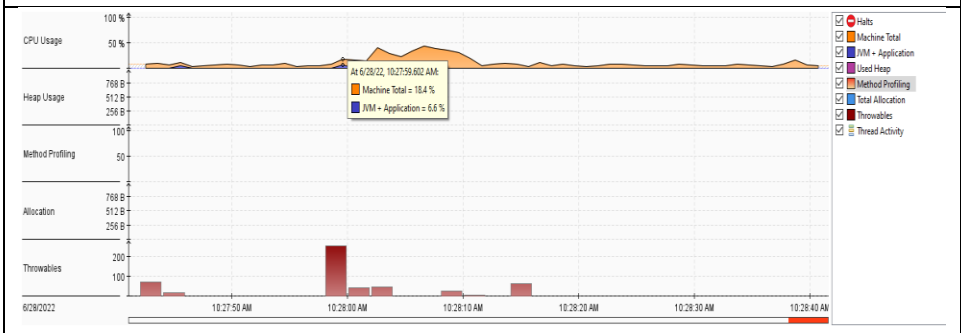
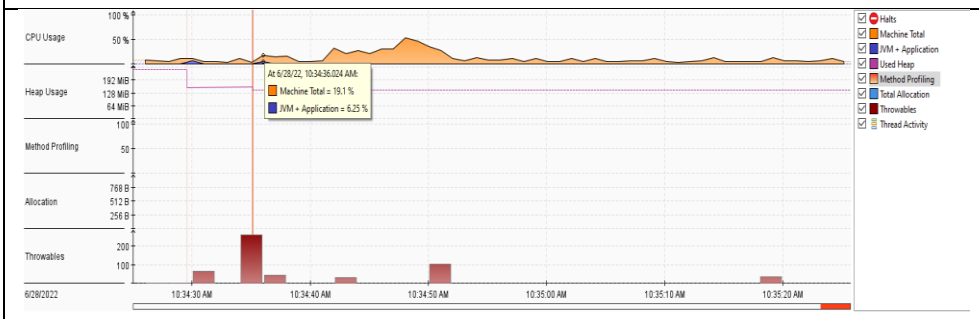
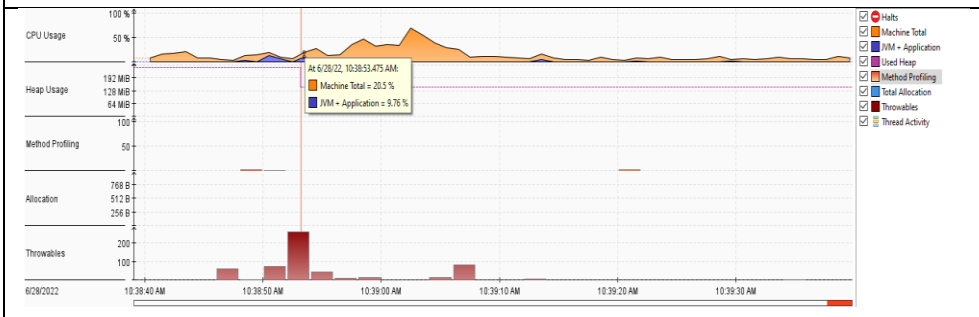
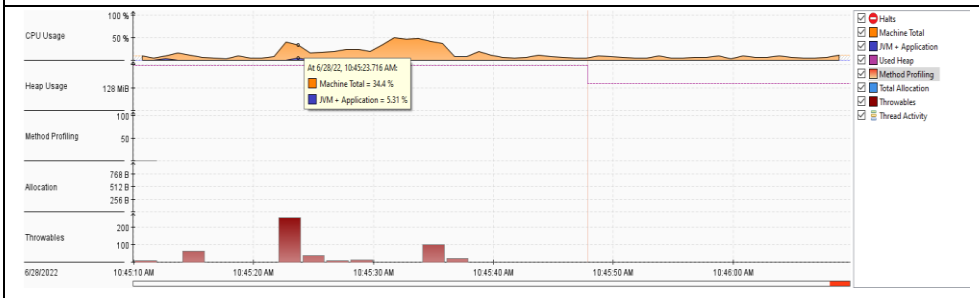
Performance Test of WebDriver()	CPU Usage
	15,3%
	16,4%
	24,3%
	8,18% GC Phase Pause
	6,6%

Table 4 Performance Test and CPU Usage of WebDriver() for AccettaFattura

Thence the *Table 4* at run number 4 there is GC Phase Pause which means there is a Garbage Collection at that run. Garbage Collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects. ava application issues with garbage collections can be diagnosed using JFR and it is automatic. With the Garbage Collection the CPU Usage will be at the minimum. That is why we are seeing a decrease in CPU Usage at run number 4 and 5.

And then, I have run the same test case called AccettaFattura for RemoteWebDriver().

Performance Test of RemoteWebDriver()	CPU Usage
	6,25%
	9,76%
	5,31%

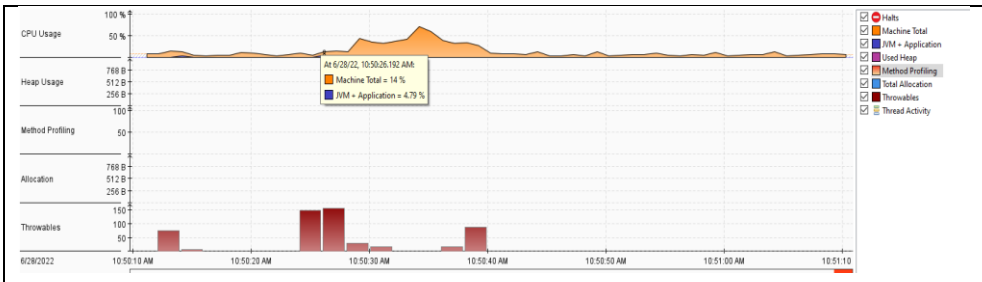
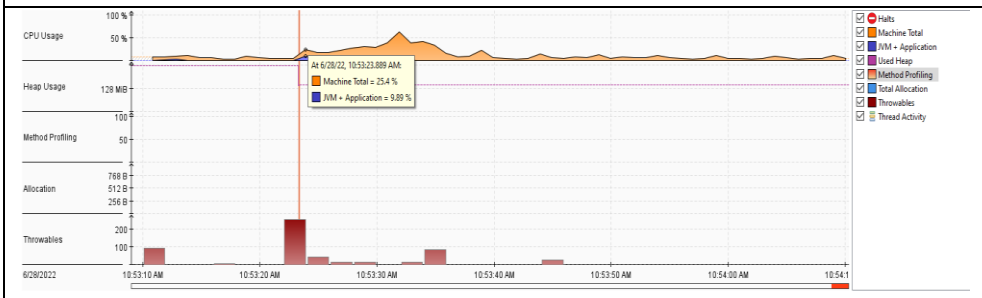
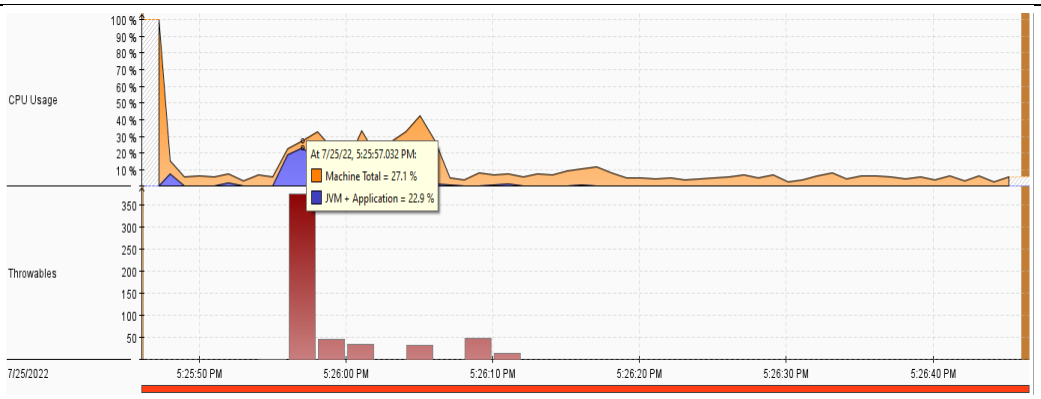
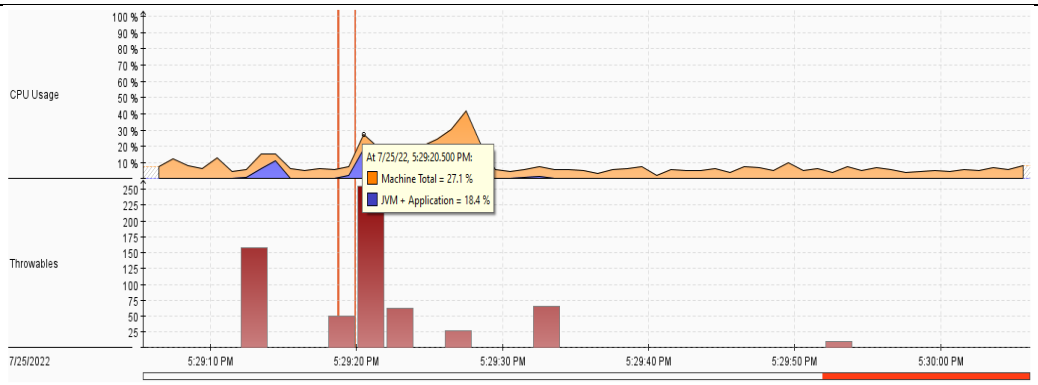
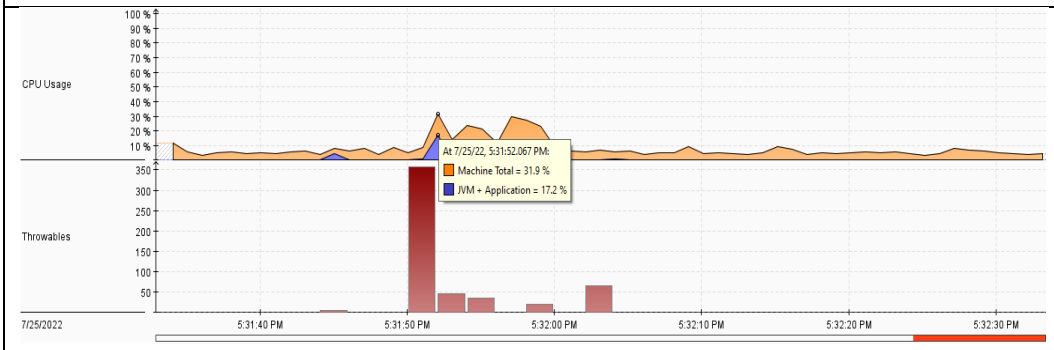
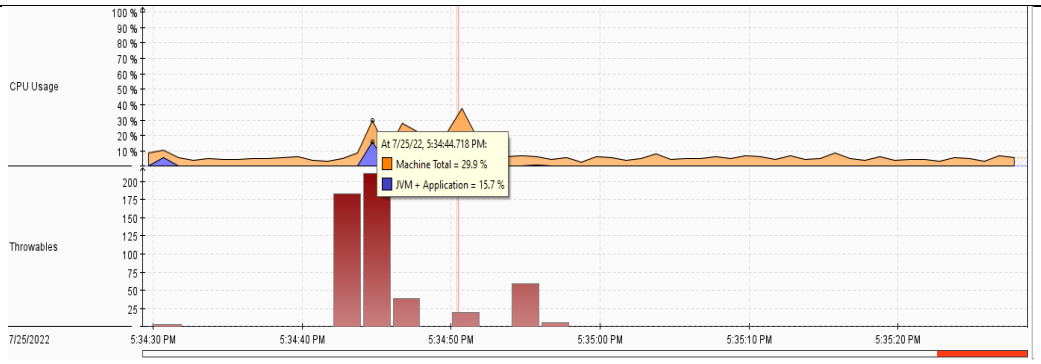
	4,79%
	9,89%

Table 5 Performance Test and CPU Usage of RemoteWebDriver() for AccettaFattura

By comparing *Tables 4* and *5*, CPU Usage is higher for WebDriver() than RemoteWebDriver(). This shows us that the WebDriver makes the computer busier than RemoteWebDriver() does. This means we can do several things together while running the RemoteWebDriver() but if we try it with WebDriver(), we might cause trouble to the computer. High CPU usage is often the cause of bad performance. Your computer may be affected by this problem if you're experiencing long loading times, crashes, or freezes.

CPU Usage is an important point for a developer who writes tests and runs them. The developer must be fast to test the code she/he wrote and imply changes faster and easier way so if every time when she/he runs the test if the computer crashes or freezes she/he cannot do the job efficiently. So, CPU Usage is a very important aspect to analyze and consider.

To have a clear idea how the RemoteWebDriver() and WebDriver() is affecting CPU Usage, I run the 'CreaSoggettiProduttori' test as I did for 'AccettaFattura', separately five times each.

Performance Test of WebDriver()	CPU Usage
	22,9%
	18,4%
	17,2%
	15,7%

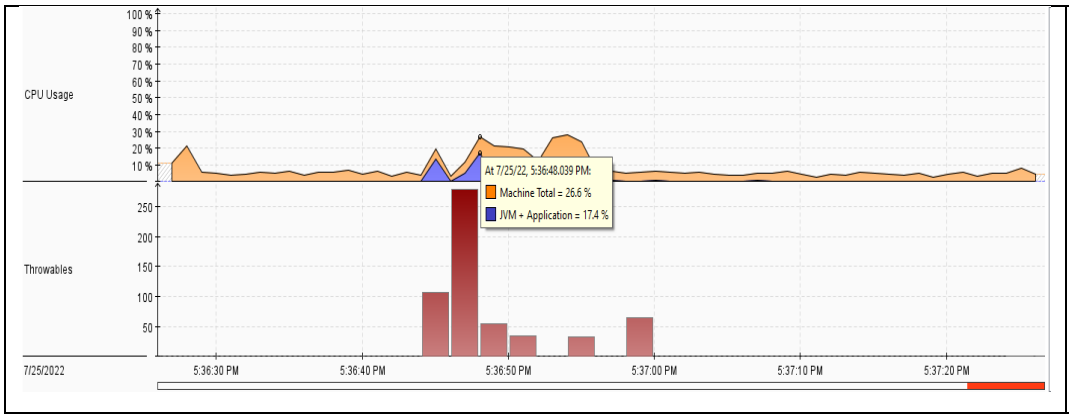
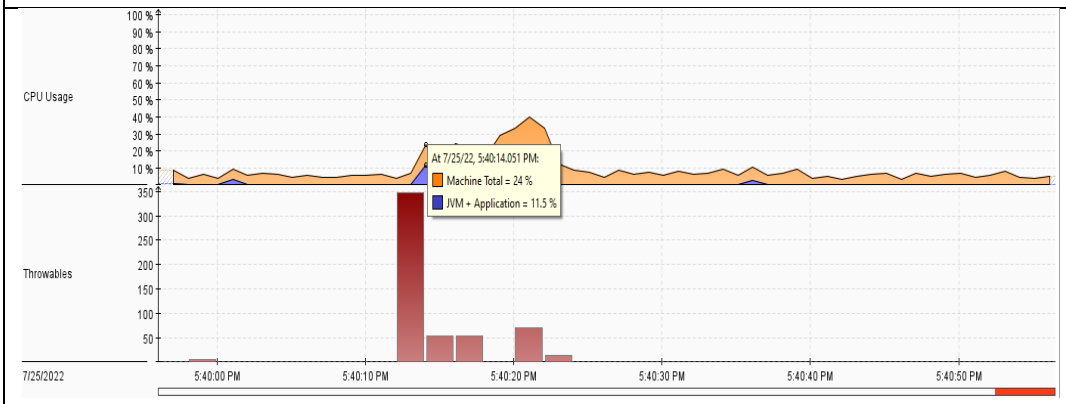
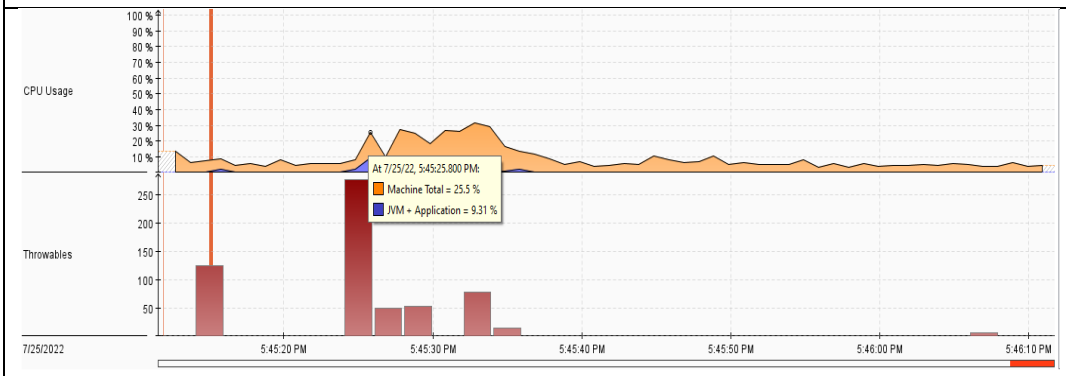
	17,4%

Table 6 Performance Test and CPU Usage of WebDriver() for CreaSoggettiProduttori

And, for RemoteWebDriver() I have run CreaSoggettiProduttori for 5 times in a row.

Performance Test of RemoteWebDriver()	CPU Usage
	11,5%
	9,31%

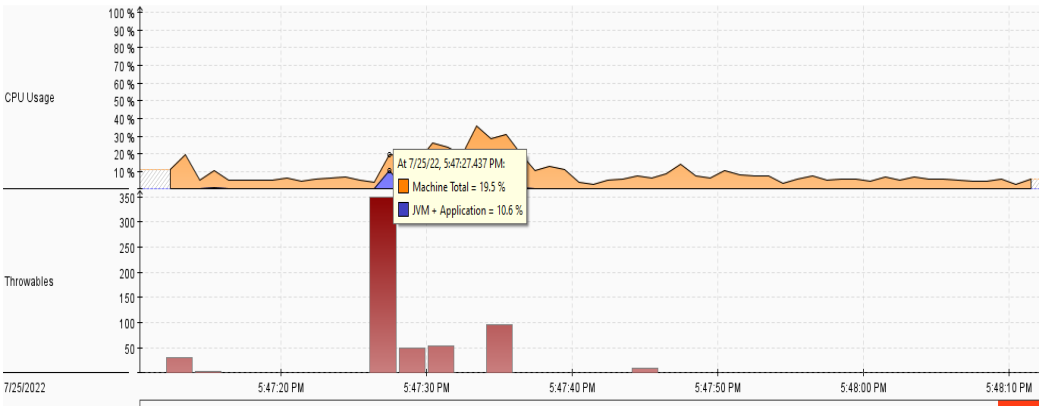
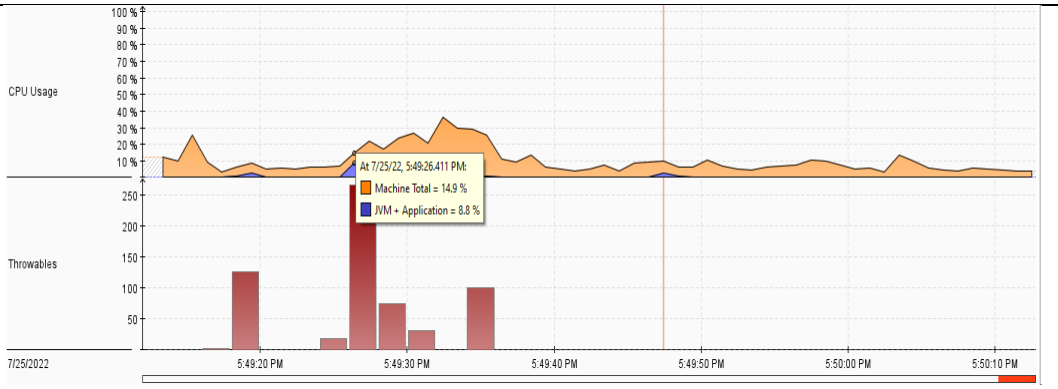
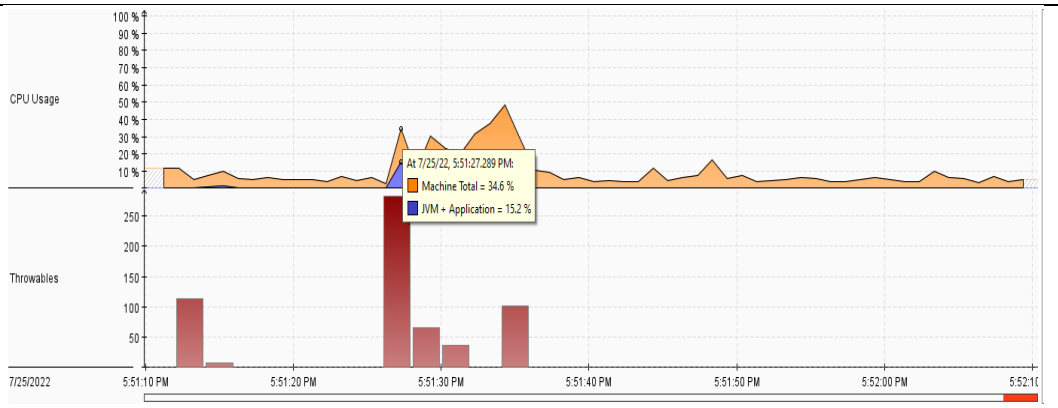
	10,6%
	8,8%
	15,2%

Table 7 Performance Test and CPU Usage of RemoteWebDriver() for CreaSoggettiProduttori

Inevitably, from the comparation of *Table 6* and *Table 7* also for CreaSoggettiProduttori class the CPU Usage is higher for WebDriver. This is a strong support for my hypothesis but also since I want to make the tests faster, I also wanted to examine the execution time. To show what is the difference between WebDriver and RemoteWebDriver() for execution time, I have run the test separately with WebDriver and RemoteWebDriver() on Eclipse and insert it into the *Table 8* below.

WebDriver()	**RemoteWebDriver**()
13.605	9.994
13.351	10.476
13.109	10.209
13.373	10.06
13.324	10.638

Table 8 Execution Time of AccettaFattura for WebDriver() and RemoteWebDriver() in seconds

In consequence of the *Table 8*, RemoteWebDriver()'s execution time is lower than WebDriver(). This result shows us that we can earn much more time every time when we run the tests. Because if we think that one test gains 3 seconds, we have 110 tests and overall, this means we can gain 330 seconds which makes the test process almost 5 minutes faster.

This amount of time makes test developer test their tests faster. To prove that again we apply the RemoteWebDriver() to another class which does not need to be improved as implementation time, due to the *Table 9* below still effective. As I did with 'AccettaFattura' I did the same Eclipse execution time experiment with 'CreaSoggettiProduttori'.

WebDriver()	**RemoteWebDriver**()
11.826	8.171
11.730	8.772
11.437	7.959
11.623	8.002
11.319	8.095

Table 9 Execution Time of CreaSoggettiProduttori for WebDriver() and RemoteWebDriver() in seconds

On account of the *Table 9* again ‘CreaSoggettiProduttori’ has very accurate results when we run with RemoteWebDriver() and WebDriver(). And, as with ‘AccettaFattura’ RemoteWebDriver() saves around 3 seconds, and this is again a very good result which even for a class that does not need to be improved for execution time.

Even though the original ‘CreaSoggettiProduttori’ is not taking too much time since it is working with the browser and windows in the browser for the future it’s very useful to make any test faster as we can.

Conclusions

Especially for big projects, performance and implementation time are important, which these applications can be used by any user at the same time. The Invoice Channel project is that kind of project and the company should guarantee to its clients that the system will not break. To do that, as a test team, we need to test faster and more cleverly and that's where I step in; during my internship, I had the task which is finding the most time-consuming and performance-lowering parts of the test cases and find a way to improve those points.

CPU Usage is an important point for a developer who writes tests and runs them. The developer must be fast to test the code she/he wrote and imply changes faster and easier way so if every time when she/he runs the test if the computer crashes or freezes she/he cannot do the job efficiently. So, CPU Usage is a very important aspect to analyze and consider.

Other than the CPU Usage, the execution time of the tests is very important for this project. One of the easiest ways to measure performance is to check the time taken by each method.

After examining the current test cases that are constantly running on Bamboo, I prepared a document to show which tests are slow and then I examine why those tests are slow, and what the common parts are of those tests.

Henceforward of the *Table 8* and *Table 9* on results, with changing WebDriver() to RemoteWebDriver() we can gain approximately 5 minutes in the testing process which means when some tests failed, we can re-run them faster. And that was our motive in the first place; to be able to run the test much faster so if we face a mistake, we can try to fix it and re-run it to see if it is fixed.

With this thesis I manage to improve the test execution time making a test case faster than before with changing the WebDriver() to RemoteWebDriver() which is easy to apply to current test cases. Also, minimizing the CPU Usage is very important for developers. As conclusion, this thesis approved its hypothesis with real life experiments.

References

[1] E. Vila, G. Novakova, and D. Todorova. 2017. "Automation Testing Framework for Web Applications with Selenium WebDriver: Opportunities and Threats". In Proceedings of the International Conference on Advances in Image Processing (ICAIP 2017). Association for Computing Machinery, New York, NY, USA, 144–150. https://doi.org/10.1145/3133264.3133300

[2] Carl Cocchiaro, Unmesh Gundecha, "Learn Selenium", Bringham, UK: Packt Publishing Ltd., 2019.

[3] P. Tripathy, K. Naik, "Software Testing and Quality Assurance: Theory and Practice", New Jersey: John Wiley & Sons, p. 634, 2011.

[4] Ifin Sistemi, "Invoice Channel® per le Fatture Elettroniche", "Invoice Channel for Electronic Invoices", 2011. [Online]. Available: https://ifin.it/.

[5] M. Sharma, & R. Angmo, "Web based automation testing and tools". International Journal of Computer Science and Information Technologies, 5(1), 908-912, 2014.

[6] K. Wiklund, S. Eldh, D. Sundmark, K. Lundqvist, "Impediments for software test automation: A systematic literature review. Software Testing", Verification and Reliability, 27(8), e1639, 2017.

[7] M. Linares-Vásquez, Bavota, G., Bernal-Cárdenas, C., Oliveto, R., Di Penta, M., Poshyvanyk, D., "Mining energy-greedy api usage patterns in android apps: an emprical

study," in *Proceedings of the 11th Working Conference on Mining Software Repositories.*, New York, NY, USA, 2014.

[8] L. Badia, A. Munari, "A game theoretic approach to age of information in modern random access systems". In 2021 IEEE Globecom Workshops (GC Wkshps) (pp. 1-6). IEEE, 2021.

[9] E. Gindullina, L. Badia, and D. Gündüz, Gindullina, "Age-of-information with information source diversity in an energy harvesting system". IEEE Transactions on Green Communications and Networking, 5(3), 1529-1540, 2021.

[10] N. Tekin, M. Kosa, M. Yilmaz, P. Clarke, & V. Garousi, "Visualization, monitoring and control techniques for use in scrum software development: an analytic hierarchy process approach". In European Conference on Software Process Improvement (pp. 45-57). Springer, Cham, 2020.

[11] T. Usländer, & T. Batz, "Agile service engineering in the industrial Internet of Things". Future Internet, 10(10), 100, 2018.

[12] A. A. Albarqi, R. Qureshi, "The proposed L-Scrumban methodology to improve the efficiency of agile software development". International Journal of Information Engineering and Electronic Business, 11(3), 23. 2018.

[13] R. Alur, P. Černý, P. Madhusudan, & W. Nam, "Synthesis of interface specifications for Java classes". In Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (pp. 98-109), 2005.

[14] P. Ramya, V. Sindhura, & P. V. Sagar, "Testing using selenium web driver". In 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT) (pp. 1-7). IEEE, 2017.

[15] U. Gundecha, & C. Cocchiaro, "Learn Selenium: Build data-driven test frameworks for mobile and web applications with Selenium Web Driver 3". Packt Publishing Ltd, 2019.

[16] E. Jendrock, R. Cervera-Navarro,I. Evans, K. Haase, & W. Markito, "The Java EE 7 Tutorial: Volume 1". Addison-Wesley Professional.2014.

[17] M. Stonebraker, "SQL databases v. NoSQL databases". Communications of the ACM, 53(4), 10-11. 2010.

[18] J. Arthur, & S. Azadegan, "Spring Framework for rapid open source J2EE Web Application Development: A case study". In Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network (pp. 90-95). IEEE, 2005.

[19] J. P. R. dos Santos, K. P. da Silva, B. P. G. Gonçalves, J. S. de Souza Pinheiro, J. M. L. de Oliveira, & D. B. de Alencar, "Selenium as a free tool to test for java web application". International Journal of Advanced Engineering Research and Science, 7(4). 2020.

[20] M. Grönroos, "Book of vaadin". Helsinki: Vaadin Ltd, Lulu. com. 2012.

[21] N. Devillard, "The eclipse software". The messenger, 87, 19-20. 1997.

[22] A. Vukotic, & J. Goodwill, "Apache tomcat 7", (p. 293). New York: Apress, 2011.

[23] T. Combe, A. Martin, & R. Di Pietro, "To docker or not to docker: A security perspective". IEEE Cloud Computing, 3(5), 54-62, 2016.

[24] B. Varanasi, "Introducing Maven: A Build Tool for Today's Java Developers", Apress, 2019.

[25] R. M. Carlson, "Atlassian: Analysis and strategic recommendation", 2017.

[26] J. Fisher, D. Koning, A. P. Ludwigsen, "Utilizing Atlassian JIRA for large-scale software development management" (No. LLNL-CONF-644176). Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States). 2013.

[27] S. M. Mohammad, "Continuous Integration and Automation". International Journal of Creative Research Thoughts (IJCRT), ISSN, 2320-2882. 2016.

[28] S. E. Seker, "Yazılım geliştirme modelleri ve sistem/yazılım yaşam döngüsü", "Software development models and system/software lifecycle", YBS Ansiklopedi, 2(3), 18-29, 2015.

[29] T. Jindal, "Importance of Testing in SDLC". International Journal of Engineering and Applied Computer Science (IJEACS), 1(02), 54-56, 2016.

[30] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep, & H. Erdogmus, "What do we know about test-driven development?". IEEE software, 27(6), 16-19, 2010.

[31] David Cohen, Mikael Lindvall, Patricia Costa, "An Introduction to Agile Methods," in *Marvin V. Zelkowitz*, College Park, MD, Fraunhofer Center for Experimental Software Engineering , 2004, pp. 2-63.

[32] Hron, Michal, and Nikolaus Obwegeser, "Scrum in practice: an overview of Scrum adaptations", 2018.

[33] F. YÜCALAR, E. BORANDAĞ, "Yazılım Projelerinde Kalitenin Arttırılması",TMMi. AURUM Journal of Engineering Systems and Architecture, 3(2), 143-152, 2019.

[34] M. Fewster, & D. Graham, "Software test automation", (pp. 211-219). Reading: Addison-Wesley, 1999.

[35] A. A. Omar, F. A. Muhammed, "A survey of software functional testing methods". ACM SIGSOFT Software Engineering Notes, 16(2), 75-82, 1991.

[36] W. Afzal, R. Torkar and R. Feldt, "A Systematic Mapping Study on Non-Functional Search-based Software Testing". In SEKE (Vol. 8, pp. 488-493). 2008.

[37] D. Nevedrov, "Using jmeter to performance test web services". Published on dev2dev, 1-11, 2006.

[38] G. J. Myers, C. Sandler, & T. Badgett, "The art of software testing". John Wiley & Sons, 2011.

[39] C. K. Sheemar, L. Badia, and S. Tomasin, "Game-theoretic mode scheduling for dynamic TDD in 5G systems," IEEE Communications Letters, vol. 25, no. 7, pp. 2425-2429, 2021.

[40] D. Atienza, J.M. Mendias, S. Mamagkakis, D. Soudris, and F. Catthoor, "Systematic dynamic memory management design methodology for reduced memory footprint," ACM Trans. Design Autom. Elec. Syst. (TODAES), vol. 11, no. 2, pp. 465-489, 2008.

[41] L. Prospero, R. Costa, and L. Badia, "Resource sharing in the Internet of things and selfish behaviors of the agents," IEEE Trans. Circ. Syst. II, vol. 68, no. 12, pp. 3488-3492, 2021.

[42] E. Gindullina, L. Badia, and X. Vilajosana, "Energy modeling and adaptive sampling algorithms for energy‐harvesting powered nodes with sampling rate limitations," Trans. Emerg. Telecommun. Technologies, vol. 31, no. 3, p.e3754, 2020.

[43] E. J. Weyuker, F. I. Vokolos, "Performance testing of software systems". In Proceedings of the 1st International Workshop on Software and Performance (pp. 80-87). 1998.

[44] Elaine J. Weyuker, Filippos I. Vokolos, "Performance Testing of Software Systems," in *In Proceedings of the 1st International Workshop on Software and Performance*, 1998.

[45] B. García, C. D. Kloos, C. Alario-Hoyos, M. Munoz-Organero, "Selenium-Jupiter: A JUnit 5 extension for Selenium WebDriver,," in *Journal of Systems and Software*, vol. 189, July, 2022.

[46] S. Gojare, R. Joshi, D. Gaigaware, "Analysis and Design of Selenium WebDriver Automation Testing," in *2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)*, 50, 341-346, 2015.

[47] H. Yang, "jmc - Java Mission Control," in *Java Tools Tutorials - Herong's Tutorial Examples*, HerongYang. com, pp. 290-296, 2020.