NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENT OF
MECHANICAL AND INDUSTRIAL ENGINEERING

MARIANA GOUVEIA DE SOUSA

BSc in Sciences of Industrial Engineering and Management

DETERMINING THE MOST IMPACTFUL BOTTLENECKS OF SOFTWARE QUALITY ASSURANCE

A SURVEY BASED RESEARCH

INTEGRATED MASTER IN INDUSTRIAL ENGINEERING

NOVA University Lisbon

September, 2024

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENT OF
MECHANICAL AND INDUSTRIAL ENGINEERING

DETERMINING THE MOST IMPACTFUL BOTTLE-NECKS OF SOFTWARE QUALITY ASSURANCE

A SURVEY BASED RESEARCH

MARIANA GOUVEIA DE SOUSA

BSc in Sciences of Industrial Engineering and Management

Adviser: Doutor André Mendes de Carvalho,
Assistant Professor, FCT-NOVA

Examination Committee:

Chair: Doutora Helena Maria Lourenço Carvalho Remígio,
Associate Professor, FCT-NOVA

Rapporteurs: Doutor José Pedro Teixeira Domingues,
Assistant Professor, Minho University

Adviser: Doutor André Mendes de Carvalho,
Assistant Professor, FCT-NOVA

INTEGRATED MASTER IN INDUSTRIAL ENGINEERING AND MANAGEMENT

NOVA University Lisbon
September, 2024

To D. Anabela and Sr. Carlos.

ACKNOWLEDGMENTS

Firstly, I want to express my gratitude to Professor André Carvalho, for the guidance and support during this process. Your advice and guidelines were truly helpful and allowed me always to know which direction to follow.

Then, I would like to thank FCT NOVA for being a home during these past years. There is no other place I would rather be a student in and if I could, I would repeat everything all over again. Additionally, I would like to extend my gratefulness to DEMI, our department, and to all the professionals part of it, that allowed me to learn, grow and understand the path I want to follow in life. Having entered the job market recently, I can see now the relevance of many things that my professors taught me. Thank you for your hard work.

Out of everyone, the biggest appreciation goes to my parents, the most important people in my life and my biggest support. Without them I would not be the person I am today and most certainly would not have been able to achieve a high level of education, which opened a lot of doors for me. If only everyone could have parents like these, the world would be a wonderful place.

Then, I need to extend a very warm appreciation to my friends, whether they are from CMB, AIESEC or FCT, thank you so much for all the love, good moments and memories I will always take with me. Life is way more fun with you in it and I am blessed to have you.

Last but not least, I would like to thank Afonso for the unconditional support and for believing in me more than I ever believed in myself during this entire process. Thank you for being tireless in helping me fight for my goals, while maintaining my motivation levels on top, which I know is a very hard task. You are a huge example.

"All our dreams can come true, if we have the courage to pursue them."

(Walt Disney).

Abstract

With the constant and fast evolution of the IT industry, companies nowadays need to focus on creating new features, while delivering a quality software as fast as possible. This makes Quality a topic of interest for software development organizations that search for the most efficient methods to achieve it.

In this sense, interviews to some IT related professionals were conducted to assess the gaps of the available literature and to, later on, create a survey. This one aimed to collect professionals' feedback and inputs on the main struggles they face on a daily basis regarding this topic.

Therefore, this investigation determined the main bottlenecks that prevent companies from achieving maximum software quality. Through the process, it was also important to understand what Quality means in this industry and through which methodologies it is achieved, as well as if companies are aware of Quality's importance, allocating enough resources and creating an environment in which it can thrive.

The completion of this study showed that the software development elements that impact the most quality assurance are testing, organizational culture and requirements, being these the ones that need the most improvement, if companies want to boost their results, maintaining high levels of Quality in their work.

Keywords: Software Quality Assurance, Traditional Software Development, Agile Software Development, DevOps, Questionnaire

Resumo

Com a rápida e constante evolução da indústria *IT*, atualmente as empresas precisam de se focar na criação de novas funcionalidades, enquanto desenvolvem um *software* de qualidade o mais rápido possível. Isto faz com que Qualidade seja um tópico de interesse para as organizações de desenvolvimento de *software* que procuram os métodos mais eficientes para a alcançar.

Neste sentido, foram conduzidas algumas entrevistas a profissionais relacionados com a área de *IT*, de forma a ser feita uma avaliação das possíveis falhas na literatura disponível e para, mais tarde, ser criado um questionário. Este teve como objetivo recolher o *feedback* e as ideias dos profissionais sobre as maiores dificuldades que enfrentam no dia a dia relativamente a este tópico.

Como tal, esta investigação determinou as maiores limitações que previnem as empresas de atingir a máxima qualidade no seu *software*. Durante o processo, foi também importante compreender o que significa Qualidade nesta indústria e através de que metodologias é atingida, assim como se as empresas estão a par da importância da Qualidade, alocando recursos suficientes e criando um ambiente em que esta possa prosperar.

A conclusão deste estudo demonstrou que os elementos associados ao desenvolvimento de *software* que mais impactam a Qualidade são a testagem, a cultura organizacional e os requisitos, sendo estes aqueles que necessitam de melhoria, se as empresas quiserem aumentar os seus resultados, mantendo altos níveis de Qualidade no seu produto.

Palavras chave: Qualidade de *Software*, Desenvolvimento Tradicional de *Software*, Desenvolvimento *Agile* de *Software*, *DevOps*, Inquérito

CONTENTS

LIST OF FIGURES

List of Tables

.

ACRONYMS

ASD	Agile Software Development.
AUP	Agile Unified Process.
CIL	Case Inconsistency Level.
DSDM	Dynamic System Development Methodology.
FDD	Feature Driven Development.
IaC	Infrastructure as Code.
IS	Information System.
IT	Information Technology.
KPIs	Key Performance Indicators.
LSD	Lean Software Development.
PBI	Product Backlog Items.
ROI	Return on Investment.
RUP	Rational Unified Process.
SCIL	Similar Case Inconsistency Level.
SCM	Source Configuration Management.
SDLC	Software Development Life Cycle.
TSD	Traditional Software Development.
XP	Extreme Programming.

1

INTRODUCTION

Software has been rapidly evolving, which is even more evident when it is taken into account that, in 2006, the software industry surpassed the hardware one, by attaining 52% of the whole IT (Information Technology) industry. Nevertheless, it is important to note that software is only a piece that is part of the entire IS (Information System) [1].

With several changes in the market, IT companies have felt the need to evolve and, to achieve that, development teams are now constantly focused on creating new features. This evolution, however, is often associated with instability for the existing systems, so that is the reason why IT operators keep being focused on software's reliability and well-functioning [2].

Having this in mind, Quality becomes one of the main concerns of IT organizations and software engineering aims to ensure it, since software engineering is a structured technical approach for quality software development and consists of applying engineering practices to fulfil this objective. It can also combine other activities, such as software testing and modern code reviews, for example [3], that will be explained further in the document.

Consequently, companies that integrate programs to measure software are the ones with projects that achieve success. This is proof for the fact that measurement is an important aspect in software development, giving an overall view of the entire project and clarity on its status regarding the selected KPIs (Key Performance Indicators) [4].

Nevertheless, the tools applied for managing projects are effective for tracking and managing their progress, but are not recommended for evaluating the progress Quality wise [5]. Therefore, it becomes relevant to study the proper tools to implement, control and measure Quality.

1.1 Primary Objectives

The primary objective of this investigation was to determine the main bottlenecks that prevent companies from achieving maximum software quality. Through the process, it was also important to understand the way people in this industry see Quality, in terms of what it means for them to achieve it in software development and through which methods, tools and ways of working they are doing it, and if companies are aware of Quality's importance, allocating enough resources and creating an environment in which it can thrive.

The study also aimed to fill in a research gap, combining all the objectives above, since there is research on several of the previous mentioned topics isolated, but not much that combine all of them, which can damage the understanding on this matter.

With the purpose of successfully completing the goals set, this project was based on the development of a questionnaire, directed at employees and organizations operating in the IT development sector, to consider real feedback and inputs of the professional who are part of the industry and deal with its challenges daily. Therefore, it was necessary to establish some intermediary goals that acted as milestones for the project development, being those the following:

- Understand the context and nature of software development and software quality assurance, through research on the different topics that are part of it and by interviewing professionals employed in IT related companies.
- Develop a questionnaire that could comprehend how quality practices are spread and applied in these organizations, sharing and spreading it within this community.
- Analyse the results of the survey and discuss them.
- Identify the most impactful barriers and limitations to the project in IT companies.

As a result of the successful fulfilment of the targets proposed above, it was expected that, with the knowledge presented in this study, IT departments and entities could have a deeper understanding of the factors that have a higher impact on the achievement of a quality software. This way, they could focus their efforts on boosting the ones that contribute the most in a positive way and improving, over time, the ones that were known to be disadvantageous. In this sense, this dissertation aimed to answer the following questions:

- **Question 1:** What is the meaning attached to quality in IT related organizations?
- **Question 2:** How is Quality structured and organized in IT related organizations?
- **Question 3:** How is Quality measured in IT related organizations?

- **Question 4:** What are the main factors preventing companies from achieving software quality?

1.2 Dissertation Structure

This study consists of five global parts: Introduction, State of the Art, Methodology, Results and Discussion, and Conclusion. The Introduction explains the context in which this study was conducted, as well as its main objectives its relevance for the IT sector. Then, State of the Art aims to understand and explain the current knowledge and research on the topics related to the main theme, such as processes, tools, methodologies, bottlenecks and culture present in the software development industry. After that, the Methodology describes the steps performed to conduct the study proposed and attain the necessary information, especially in the questionnaire developed, and the chapter of Results and Discussion explains, analyses and discusses the main findings of the project developed regarding Quality in IT companies. Finally, in the Conclusion is presented the main findings of the study, as well as the main barriers and limitations that the implementation faced along its development.

Besides these, there are some other sections that serve as an assistance for the rest of the document, being these the Bibliography and the Appendix, where all the literature and supporting information can be found.

In resume, this document was built based on two different parts, where one is more focused on the existing literature and theory, and the other is more centred on the analysis and discussion of the results obtained, with the Methodology section acting as a bridge between the two of them.

2

STATE OF THE ART

2.1 Software Development

Software appeared around 50 years ago, starting as a very different process from what it is known nowadays. In the beginning, software development was simply known as coding and then fixing the code, being this a process that did not require a lot of planning and was easily influenced by decisions taken now. However, as everything progressed, it became more difficult to work with [6].

More and more there is a request for flexible and, at the same time, rigorous procedures to develop software that can not only follow the fast pace at which changes happen, but also ensure the result's absence of flaws and predominance of Quality [7].

The process that aims to develop software is the SDLC [6], [8]. This process can also be seen as a collection of the needed assignments to create software with Quality [9], [10], being these: "requirements elicitation, design, implementation, verification, maintenance" [10]. Usually, the IT department is focused on quick deployment of code, which means, rapid implementation of new functionalities [11].

Therefore, software development is considered as a complicated duty with multiple restrictions in the way of its achievement [12] and there is an increase in software testing to ensure the outcome is of the highest Quality [13], always with the thought that software quality "means different things to different people" [14].

2.2 Quality Assurance and Management in IT

The degree to which final products fulfil their requirements, as well as their overall performance, is what it means to be in the possession of Quality [11]. In the SDLC, quality assurance is seen as one of the most indispensable pieces [15].

In the literature, there are multiple definitions for SQA (Software Quality Assurance), but what all of them have in common is that it means to be in conformance to certain defined procedures. On the other hand, when it comes to software quality management, it ensures that these processes are being well established and later followed [15].

To successfully develop software, there needs to be awareness about SQA. This necessity is also proven by the fact that the success rate in software projects in 2018 was around 37%, besides implying that further quality assurance efforts should be placed during software development [13], as its success is measured based on the software quality [15]. The software industry progress in terms of success rates did not change drastically as expected through the years, given the fact that, in 1996, the success rate of software projects was 33% and, in 2008, 44% [1].

A study in 2003 showed that between 60% and 80% of software development projects take longer than expected to be fully completed. Besides this, several sources from both academic and empiric natures claim that, in this industry, working overtime is also a common practice [16]. The same study also showed that around 60% of the efforts put in software engineering were spent on finding and fixing software defects. In fact, after the software is released, its defects cost between 40 up to 1000 times more to fix [14], [17], which financially speaking represents trillions of dollars spent in IT projects all over the world and another set of trillions of dollars spent on IT projects failed [18].

In software development, defects usually exist and are translated as errors written in the code [11], [14], since this is written by humans susceptible to flaws, being this the primary cause for these defects [14]. Some defects that appear can be divided into seven types, being them clarity, completeness, compliance, consistency (if there are mutual exclusive requirements, for example), correctness, testability (if it is passive of being tested, through the way it is written) or other (any defect that may not be included into any of the previous types) [19]. Errors are a threat to the trustworthiness of the software, being also a financial threat, as finding and fixing them [20] has been estimated in a study conducted in 2018 [14], [20], to charge companies with a high monetary value [14].

There are several quality assurance activities that can be put in place along the development process, such as "planning, defining and implementing quality standards, reviewing and auditing, testing and reporting on the outcomes". These activities carry with them multiple benefits from different natures, such as financial, Quality wise, productivity wise and even improving the satisfaction of the final user [13].

Additionally, a software development project that is oriented to Quality can be fulfilled through the usage of quality characteristics, such as security [5], maintainability [21], functional suitability, reliability [5], [11], the time it takes to change the request [11] or efficiency [21], completely disregarding the type of methodology applied to complete the process. But even with the existence of these metrics, assessing Quality is a challenging process when the software is still under development [5], [22].

Nevertheless, despite the efforts to guarantee software quality, projects continue to fail [1], [15] and damage the reputation of enterprises [7], as it is complex to measure software quality and to understand how each project may influence the metrics differently [22]. As an attempt to improve this situation, even new approaches for software development continue to be created [23].

Software testing is a common implemented procedure that ensures software quality [15], since the quality of the procedure is a directly related factor to the quality of the results produced after it [9], [15], [22]. This means that the improvement in the process translates to less bugs in the final software as well [22].

Activities to control Quality are essential in software development [12] and, accordingly, there is a growing reliance on testing [13], since it is a common procedure that ensures software quality [13], [15], [24], despite also being is one of the activities that spends more resources, whether they are financial or time wise [24].

Testing consists of putting the software working under certain defined conditions, to first guarantee it works as it was intended [8], [24], then to understand if there are any bugs, errors or functionalities missing [6], [8] and lastly to ensure it fits the desired purpose [6], [24]. Sometimes it requires different approaches in terms of technique or framework to make sure the process is fast and has high quality [25], given the fact that not every methodology is appropriate for everything that is developed [8]. So, testing tools are an assistance for developers to evaluate the software in terms of bugs and guarantee its reliability [24].

The main point of this process is delivering products with quality and that is the reason why this activity is a big part of quality assurance processes. The decisions it is associated with,

for example knowing what is going to be tested, are important, since they can influence the amount of bugs detected [8].

The advantages of testing software include "reusability, repeatability, and effort saved in test executions" [24]. The implementation of a continuous improvement process guarantees that Quality also benefits improvement wise [26]. Adding automation and incorporating more testing activities ensures that quality requirements can also be covered and not only the functional ones. Also, the definition of criteria for testing activities allows quality attributes to be easily used as KPIs for Quality that can be tracked daily and give an ongoing status [5].

Some years ago, testing was considered a secondary activity, with an engineers' majority, in 2015, claiming they did not like it and that they would not choose being a tester. Furthermore, around 2013, testers were not receiving the same attention as developers in Agile projects, since this last group was seen as more important [27]. Nowadays, and this has been increasing, testing activities determine the quality of the software delivered [8], [14]. It is an expensive [6], [24] and important part of the software development procedure [8], [25], because 40% of its costs are towards this [6] and if there is not enough testing, there can be harmful consequences for companies [27]. Therefore, it is important that there is a balanced relation between Quality and costs, when it comes to testing [8].

Therefore, and particularly in Agile, companies now recognize the importance of testers [27] and have them participating since the beginning of the process, since they are responsible for a key activity that occurs continuously during the Sprint [6].

When not associated to software, being a tester means being "a person or machine who assesses the quality or state of a thing by testing it", "the ones responsible for carrying out testing and building up test cases and test plans" or "professionals who identify new defects from failed test cases, analyse defects, and report them in a bug-tracking system". In reality, testers are known for doing a lot more things than what is usually described and these come with requirements for a certain set of skills. These skills used to be communication, problem solving and analysis, but others like adaptability, autonomy, receptiveness to new ideas and being a team player are becoming more valuable, given the growing need to focus on ethics, the customer and working under pressure. Nonetheless, there is still a strong inclination towards testers that have a vast knowledge in practice. Regarding technical knowledge, testers are expected to design the tests and take on tasks related to performance tasking, but also related to software development. According to a study taken in 2019, "testers need to master development better than developers need to master testing". All of these make this group

work as quality assurance and have responsibility for parts of customer support or project management [27].

Software testing has four levels: Unit Testing, that refers to an individual module being tested [8] and is usually referred to testing brand new code [28], Integration Testing, that tests a group of modules working together and their compatibility, System Testing, in which the whole system is tested and errors are checked, and User Acceptance Testing, that aims to test everything from the user's and consequently, the real world's perspective [8]. From all of these, the last two are the most esteemed when it comes to industrial demand [27].

The User Acceptance Testing is a combination between Alpha Testing and Beta Testing. During the Alpha Testing, the application is being checked from the developers' side by the developer, it can be through the White Box or Grey Box techniques and when it is concluded, it is a signal that there are not any other features being included or changed. After Alpha Testing is concluded, there is Beta Testing (also called Formal Acceptance Testing), in which everything is confirmed from the users' side by the user and a developer, so that improvement points and general feedback can be suggested. This is why when the application is at this phase it is considered a prototype, and the ultimate version is released after this testing [8].

Then, there are three basic software testing types: White Box Testing, Black Box Testing and Grey Box Testing. The White Box Testing is effective and refers to testing the functional part or the internal structure, meaning that it checks the logic inside the application. It can also be called Glass Box or Structural Box Testing and it is possible to use in any of the first three levels of testing mentioned before. On the other hand, Black Box Testing confirms the application's functionality, through tests that include functional variances, and does not care about any internal work. This strategy is widely used worldwide. Finally, Grey Box Testing is the combination of the previous two: Black and White Box Testing [8].

On top of everything so far, there is an indispensable parameter: testing metrics, which its main goal is to acknowledge the software parts developed that require exhausting testing. These metrics are responsible for showing whether testing is being successful regarding the desired results and provide constant focus to improve the process [8], [24], also highlighting the areas that need improvement [8]. Some examples of testing metrics are organization, project, process, product, static and dynamic metrics (the static metrics are computed, but do not need executing code, yet the dynamic metrics require this part) [24]. The "Average Defect Response Time", that demonstrates operational efficiency by the average time the testing team takes to identify and respond to a defect is one example [8].

There is not any standard for a Software Testing Life Cycle, so it depends on each application. Therefore, an example of a basic structure for it could be the following: Requirements Analysis, Test Planning, Test Designing, Test Case Execution, Test Result Reporting, Defect Retesting, Regression Testing and Test Closure. For this process, in the first step there is an analysis from the quality assurance side on the software requirements to ensure the fundamental conditions are taken into consideration. Then, there is the Test Planning for the strategy definition, which is the principal activity of the process, and after that there is Test Designing to develop test cases that consist of a set of inputs based on data that executed under certain conditions should result in some conjectured results. Based on these, the function being tested can fail or pass the Test Case Execution, so the function can, respectively, carry or not carry defects. The Test Result Reporting is made with a bug/defect report and, in case there were defects found, there is a rectification from the development team's side, so the Regression Testing phase checks the corrected features. Later on, when the testing cycle finishes, the software release cycle begins [8].

2.3 Traditional Project Management

As mentioned previously, software development started as a disorganized process. Despite working for simpler and smaller systems, with more complex systems appearing, there was a need to develop a methodology for better software development. This way, there was a disciplined method that led to more effective and efficient result and a list of steps was created, being these: requirements [6] or planning [11], analysis, design [6], [11], development [6] or coding [11], testing [6], deployment and maintenance [6], [11]. It initiates with the collection and documentation of all the project's requirements and, after that, proceed to the design and development [6].

The traditional methodologies consist in executing separately all these steps [6], [9], [11], with the options of leaving testing and deployment towards the end [6] or just deployment [11]. The first case leads to a misidentification of errors during the process, but to try to avoid these, there is a detailed strategy and desired outcome understood at the beginning and then followed thoroughly [6]. This means that Traditional Software Development (TSD) focuses more on a thorough planning and documentation, with a broader view in terms of design for the projects, which through time made these methods not compatible with the digital evolution [6]. The SDLC is in constant change and currently does not focus on its process'

performance, but rather on iterations and the connection there is between the process and the originated value it can offer, hence why new methods appeared [2]. However, traditional methodologies for software development do not adapt themselves to the struggles that arise with the digital transformation, giving their attention to detailed plans [6], [10], defined processes [10], large-scale design [6] and rigorous documentation [6], [10]. The challenging part of this last topic appears whenever there are cultural differences or language barriers among people inside the same project [10].

Some examples of approaches considered part of TSD are: "Waterfall, V-Model and Rational Unified Process" (RUP) and, due to the way their processes are built, these can be called "heavyweight" methodologies [6], [29] or "structed processes" and are planned to contain bigger teams [10]. Any of these is an appropriate method for projects with predictable requirements and unchangeable targets [10], [14], since it provides bigger control over the process [10]. In any case, Waterfall is one of the oldest and most known approach from the ones presented [30], suggested in 1970 by Winston W. Royce [29], [31], where the output from one phase is used as an input for the next one and hence the name of this approach [30].

Also, project managers very used to traditional methods did not appreciate the new ones, since their teams needed to keep adapting the code to new changes, which only led to having them working more. This happened, because of the lack of more effective testing approaches and gave the impression that teams were incompetent for, in their perspective, constantly executing changes that were not contributing to the final result's improvement, despite the reality [32].

2.4 Agile Project Management

During the 1990 years, ASD (Agile Software Development) gained popularity due to its focus on customer satisfaction and business or information system relationships [11], but Agile as it is currently known only started in 2001, introduced by the Agile Manifesto [33]. This is a collection of values and principles that has the goal of improving the software development process [18] and every methodology, to be considered Agile, must follow the same ones [2], [18].

This alternative to traditional methods appeared, because of their weaknesses [9], such as the lack of opportunity for regular changes in complex projects [7] that were making project management difficult, thus existing the need to overcome them [9].

Another one of the main differences between these two types of software development methods is that in Agile there is a team approach, and everyone is responsible for guaranteeing product quality [7], so every team member takes part in testing activities or related [6], [7]. Additionally, the entire process already contains quality assurance procedures without the need for an explicit quality related role [7].

All these characteristics require organizations to change in terms of mindset, culture [9], [18] and structures [18]. They need to be ready for iterative software delivery [6], [34], for training their teams and for setting the business solution if they want this software development process to thrive [6].

At the moment, it is a set of methodologies for software development [6], [22] that targets the solution with a much lighter and quick process [22]. The main features that compose Agile are smaller [10] and self-disciplined teams, responsible for organizing their own work, quick execution, being driven by value and oriented to business [18], meaning that it does not take into account time consuming activities, given the possibility of having them decrease the cost of the final product and spending resources in general [35]. This way, Agile enables project development's five dimensions: personal, organizational, technical, project oriented and process oriented [13].

Agile is based on a gradual delivery through incremental additions [6] leading to continuous iterations [2], [6], [7], [27], testing [7] and planning [6], as well as continuous software delivery [5], [9] instead of doing all of this closer to the project's ending [6].

Each delivery happens within a short period of time [6], [9], to guarantee that the customers can start participating in the process [2] soon and that the changes imposed by them are implemented right away without damaging the process stability, giving the developers the proper time to execute these changes [6].

That is the main reason why one of the proposed definitions for Agile is "an iterative lifecycle designed based on short delivery cycles to deal with uncertainty in scope and rapid change in requirements" [18].

Agile project management has become popular in software engineering [13] and many organizations adopt it so they can improve IT projects success rates [18], with an adoption rate growing from 59% to 83%, respectively, from 2011 to 2012 [22].

There are several Agile methods, such as "Adaptive Software Development, Agile Unified Process (AUP), Crystal Methods, Dynamic System Development Methodology (DSDM), Extreme Programming (XP), Feature Driven Development (FDD), Lean Software Development (LSD), Kanban, Scrum and Scrum ban" [18]. Out of all of these, the two most implemented are Scrum

and XP [2], being Scrum the most famous [7], [22] and adopted [18], with an adoption rate of about 75% [14].

2.4.1 Quality Assurance in Agile Project Management

The previously mentioned characteristics make Agile's process more driven towards value [2], [18] rather than planning [18] and more dynamic, including continuous feedback, continuous learning within the team, besides improved communication and collaboration [6], [9], [13], as well as ensuring that everyone is working towards the same purpose [6]. Activities than enable quality assurance are inserted in the team's daily work, so that the quality of the final product can be improved as smoothly as possible [7].

Other benefits of Agile are the reduction of risks, because of effective changes management, transparent reporting on the product demonstrations and the perfection of the process through the discussion that happens [6]. The collaboration that comes from the customers during Agile Software Development [6], [14] ensures motivation within developers, improves communication, as well as the understanding of customers' problems, since it comes from an end user perspective [14]. It also enhances the visibility they have on the company's processes [6], [13], [14] and builds trust on prioritization decisions [14], so even with time pressure [13], [16] and restrictions in financial resources [13], Agile can deliver last minute requests [13], [16]. Agile enhances systems' quality [6] and productivity [6], [36], through the focus on this last topic and on the completion time's reduction [9]. In any case, Agile's success is dependent on whether the team can continuously improve the final product until its costs and time to market are reduced, while preventing the software system from becoming too big or complex [22].

Nevertheless, the previously mentioned collaboration factor between stakeholders and team has some bottlenecks attached, because if this involvement is not properly made, it can lead to damaging consequences for the project. Besides the fact that customers are unpredictable and that they do not possess pertinent expertise on the subjects, there is also the issue of projects in which the final amount does not change regardless of the hours worked, leading to a pressure exerted on the team to work beyond expected, resulting in a productivity drop and loss of business opportunities [14]. Regarding additional bottlenecks, there can be some delays when giving customers new features, incompatibility between prior and new software features, lack of quality before the product is released and not fulfilling budget or deadlines [9].

Overall, large organizations often struggle with combining Agile requirements, so basically flexibility, collaboration and transparency, with following quality assurance procedures and ensuring the quality of the developed product [7].

In terms of metrics, there are several used for Agile projects [18], [22] with focus on requirements' quality and time related [22], like customer satisfaction, defects resolution or cycle time [18], even though there are not any for tests' quality [22]. It is also confirmed that there is a continuous improvement of Quality through the definition of the metrics to be achieved in each phase [5]. Furthermore, if "number of defects" is considered as a software quality indicator, then Agile methods represent a bigger software quality, as Agile can increase a team's chance to attain Quality [14].

The implementation of this type of practices in general leads to better performances, less overtime work, employees knowing how to deal with stress or feeling less tension and more customer satisfaction [16].

2.4.2 Scrum

Scrum was created in 1986, firstly mentioned by Takeuchi and Nonaka on a paper written by them, with the goal of introducing a more flexible development process. However, the term "Scrum" and the methodology currently known was only invented by Sutherland and Schwaber and later on applied in 1995 [7]. Scrum's creation had to do with the reasoning that a software development project could not be planned entirely in the beginning, as it is something too volatile and unpredictable, with plenty of changes needed during the process. To make this possible, Scrum is iterative, so it develops software based on multiple iterations [7], [18] rather than in a single release, making it possible for changes to be better managed. At the same time, the need for constant change and adaptability should not interfere with Quality, so it is also in Scrum's best interest [7].

Scrum is considered a framework that can incorporate several other techniques and processes to solve more structured problems [35], assisting in the development of complex products as well [12]. This framework is based on three roles, which are the Product Owner, the Scrum Master and the Development Team [7], four events [12], such as the Sprint Planning Meeting, the Daily Scrum, the Sprint Review and the Sprint Retrospective [7], [12], and two artifacts, that are the Product Backlog and the Sprint Backlog [12], [35].

The three roles are part of the Scrum team, organizing themselves and being multi-sector, focusing on repetitive production and gradual product delivery to ensure there are moments for feedback as much as possible [35].

Product Owner: has the highest level of knowledge about the client and their interests for the project [35], so is responsible for representing them [7], [12]. This person oversees managing and defining priorities for the items in the Product Backlog [12], [22], [35], guaranteeing it is clear to everyone [22] and for making sure that the rest of the Scrum team understands the requirements for the project [35], also leading them towards the next steps [22].

Scrum Master: consists of a coach to the Scrum team when it comes to properly following the Scrum process [7], [12] and the strategies implemented by them may determine success [35]. This person is responsible for aiding the team when it comes to the creation of high-value products, removing all the possible impediments and ensuring a suitable environment when achieving the goals of the sprint [7], [22], as well as helping the Product Owner to manage the Product Backlog and prioritize items in it, with the goal of maximizing value [22], [35].

Development Team: varies from 3 to 9 people and it's a cross-functional group [7], [12], [22], meaning there is a variety of skills [7], [22] within that can go from programming or testing to analysing [7]. Some refer to the team members as developers [22], since the team members are responsible for developing a possible addition to the final product in each Sprint iteration [12], [22] and confirming if it is according to the desired criteria [35].

Sprint Planning Meeting: determines the beginning of a sprint [7] and gathers all the team members [35] to determine the Product Backlog items that will be addressed during the discussed Sprint [7], [12]. The Sprint goal is chosen, also serving as identification for the reason behind it [12], and the work that is going to be done is planned [7], [12], [35], through the choice of items from the Product Backlog to integrate the Sprint Backlog [12], being this a meeting that can take up to 8h when it is a one month Sprint [35].

Daily Scrum: happens every day through the Sprint [7], [12] and its purpose is that every team member answers three questions ("What did you accomplish yesterday?", "What will you commit to completing today?" [12] and "What impediments stand in the way?"). This allows the team to understand the progress being made [7] and to resolve as soon as possible the existent impediments, if possible, or to escalate them to the Scrum Master for the same motive [12], maximizing the chances that the Sprint goal is achieved. This meeting only takes up to 15 minutes per day [7].

Sprint Review: meeting that takes place whenever a Sprint comes to an end [12], [35]. The Development Team presents to the client the work that has been done [12] and thoroughly assesses its outcome [7], [12], [35]. The Product Backlog is reviewed, and some items might be added [7], [12] before the next Sprint Planning Meeting [7] in order to better define what to do next [12]. Usually, it takes a 4h meeting for a one-month sprint [7], [35] to complete what was previously mentioned, to collect feedback [12] and assess the knowledge gained during the Sprint [35].

Sprint Retrospective: the last meeting of a Sprint [7] that takes place after the Sprint Review, but before the Sprint Planning Meeting [12], [35]. There is an evaluation of the work that was done [35], with the identification from the whole team of the points that went well and the ones that need improvement [12], in terms of the team [7], [35] and the project in all its components [7]. The main purpose is to identify improvements and strategies to be implemented in the next Sprints [7], [12], so that there is refinement continuously during the process [12].

Product Backlog: contains the user stories [12], [22], [35], i.e. an explanation in three or five lines [35] of the software feature from the end user's point of view [12], written by the Product Owner together with the stakeholders [12], [22]. These stories are used as requirements for the project being developed [12], [35], designated by PBI (Product Backlog Items) [7] and organized by priorities [22], [35], according to their business value [7]. It is an artifact that evolves through the process, due to software development requirements being dynamic [35], which leads to the change in problem understanding and context [7].

Sprint Backlog: group of items selected from the Product Backlog to be completed during a Sprint [12], [22], [35]. Usually these are the items with the highest priority in the Product Backlog that later are put here [22]. There are not changes to the items during the Sprint, unless it is really necessary and then there is a restart for the Sprint made by the Product Owner [35].

To start the process, initially the Product Backlog is built, and its items are prioritized, so that, after this, the entire team can create the Sprint Backlog, through the estimation of the number of items that can fit into a Sprint. After that, the Sprint starts, taking usually between 2 to 4 weeks and the process continues with the Scrum team developing, designing and building the chosen features [22].

Scrum is associated with a lot of potential, when it comes to increasing software quality. However, this is not a replacement for proper software engineering, quality assurance and

quality control practices, but it is rather a complement given the nurturing it makes to the environment of the team [14].

It is known as one of the best Agile processes due to its contribution to quality assurance [13] and some of its benefits are the fact that it is simple and easy to implement, as well as able to amplify teams' productivity [14]. It makes the last-minute changes to the customer's initial requirements easier without discarding Quality and the technical excellence [13], even with testers inside the team, since this could make the developers less accountable for code quality, as it happens in other processes [14]. Additionally, because it is based on several iterations, it enhances the predictability of the result and diminishes possible risks [7].

Still, despite the benefits presented and the possibility of this methodology to adapt to modifications, there are still problems related to quality in Scrum, mainly associated with process and product [7], meaning that not always adopting Scrum or other Agile methodologies will be a solution to improve Quality [14].

Particularly about Scrum, there can be some bottlenecks in achieving software quality if its implementation has inconsistencies, if there are cultural constraints or even if there are tensions amongst teams. Some examples for the last one may be members following their leader's orders blindly and not speaking their minds with the fear of repercussions. Another issue that may arise is related to the length of the sprint, since if it is too long, then there is the possibility of a misalignment between the features developed and the business needs. All of these contribute to Scrum teams not improving the quality of the work they develop [14].

All of these characteristics make Scrum easy to understand, but significantly difficult to become an expert in [35], since even in companies used to its difficult to apply quality assurance during the process [7].

2.5 DevOps

DevOps appeared in the early 2000s [9], [37] thanks to Patrick Debois [2], [9], [37], [38], at a time when it was common to exist a separation between developers and operations [37]. This resulted in conflicts between both teams regarding their goals, inadequate environment for testing and a flawed information flow [38]. Consequently, these traditional software development methodologies were becoming challenging to use and were not addressing entirely the needs of the industry [37].

After hearing and being inspired about related topics from other professionals, such as Paul Hammond and John Allspaw in conferences and lectures [9], [37], in October 2009 [9], [11], [37], Patrick Debois decided to organize an event named DevOpsDays that aimed to address the big gap between development and operations [2], [9], [37]. It gained so much attention from professionals of both sides that it made some smaller IT companies start right away to implement the tools and practices associated with this new methodology [2], [37], which was now shortened to DevOps [2], [11], [37].

Nonetheless, it was only in March 2011, when Cameron Haight outlined his projections for DevOps, that there was even more attention being drawn to this concept and all size IT companies started to implement the new method. Since then, it has been known as a big development for the IT industry like Agile had been some years before [37].

The main purposes of DevOps are, firstly, to accelerate the development process [11], [39], through several factors, such as tasks' automation [9], [11], continuous delivery of software [2], [9], [11], [38] and improved communication and collaboration among teams [2], [11], [38], and secondly, to still deliver software that is reliable and high quality [9], [39], by increasing the stability of the developed software, which contributes to the customer satisfaction [11]. It also diminishes software development costs, since it creates a more productive software delivery process [38].

According to different articles, there is a divergent understanding about the pillars, principles and characteristics associated with DevOps.

DevOps can be based on six principles: customer-oriented actions, considering options with the final goal, responsibility from the beginning to the end, multifunctional autonomous teams, continuous enhancement and automation as much as possible. There is a focus in automation and feedback due to the goal of reducing failures and improving communication amongst team members, which directly leads to the improvement of software development quality [13]. On the other hand, although related, there are some important factors associated with DevOps' success, which are the people and the culture. These are the main factors since they include the ongoing feedback, participation of the customer in the process [40] and a cultural shift [41]. Nonetheless, DevOps' key characteristics can be considered as "automation, collaboration and quick services", while the foundation layer for it are four pillars: "culture, automation, measurement and sharing" [39]. This foundation layer can also be considered including one more area, which is lean [2], [9]. DevOps can still consist of three principles: flow, feedback and continuous learning. The first one refers to the teamwork existent between development and operations, the second is about the usage of values from Agile and the third

one indicates that there are ongoing successes and failures that contribute to the team's learning [11].

There is also no agreement about the definition of this term until now [9], [41], with DevOps being considered ambiguous [38], so there is a variety of possible meanings for it: modern software development method [6], change in IT culture, infinite loop of operations [38], group of principles [9], [41], set of agile practices [9], [39], cultural movement [39], collection of engineering related practices influenced by cultural aspects [42], etc. The only point in common is that it implies a collaboration between the two previously mentioned groups of people. The reasoning behind this could be the innovation and the greenness of the field or even the fact that DevOps started to be used in practice and only after it the research started [41].

In this sense, there is the suggestion to define DevOps as "the interaction between development and operations" [41], or even to leave it without a definition, for which both options aim to leave people the freedom to best define the term according to their context or even to let each author tailor the definition according to their approach [9], [41].

So, as already discussed, there is no consensus about the definition of the term DevOps [9], [41]. Likewise, there is no consensus about the practices that should be adopted in its lifecycle [41] or the roadmap for its implementation [9].

Some of these practices could be continuous integration, continuous deployment, continuous delivery, automated testing, communication encouragement and management, sharing responsibilities, automated deployment, continuously evolving skills with continuous learning and experimentation, sharing knowledge, continuous testing, having different environments, infrastructure as code (IaC) [2], source configuration management (SCM), defining new roles and teams or even modular architecture [41]. Out of all the possible practices, continuous integration, continuous deployment, continuous delivery [5], [9], [38], [41], continuous testing and continuous monitoring are the activities more accepted within DevOps execution [38], [41]. Continuous integration improves the system quality through the continuous merge of code, while continuous delivery increases the speed for software delivery, due to the automatic code release to the testing environment [6].

Unlike TSD, where the deployment is only done at the end of the software development life cycle, DevOps wants to provide new software features more frequently, so the deployment is usually ongoing, multiple times a day even [6].

When it comes to process, it usually follows a sequence of seven phases, which are planning, coding, verifying, testing, deploying, operating and finally, monitoring [5], [38]. The first

phase is meant to define the project's requirements and task management, then through coding these requirements are put into practice isolated and in the building stage they are combined, so that the complete application can start to be built. After everything is put together, the application is tested so it can be deployed after and tasks related to management can be performed, ensuring continuous service, with the final step of monitoring all the information from the users' perspective and operations [5].

Whenever a developer submits code to the code repository, there is a triggered script that tests it automatically, giving feedback right away on whether there are errors and, in case there are not, stores the code. With this storage completed, there are two options: another script can be triggered to deploy it in a testing environment together with new code, so that testers can have it available for them; or a more elaborated and automated testing can be done, to keep developers accountable for new code failures. This is what contributes to accelerating the process of fixing bugs and, consequently, increasing stability in the software [2].

Around 90% of the companies mention that customer value improved with the adoption of DevOps [11], which means that it has great potential, despite its current lack of definition in many aspects [39] and low adoption rate [9].

In terms of positive aspects, DevOps promotes and enhances the collaboration between development and operations [9], [38], [39]. Also, it allows freedom for failure, as it is run by the motto "The faster you fail, the faster you recover", giving the opportunity to experiment different outcomes, which can trigger new products or services and increase customer satisfaction [2]. Other benefits associated to DevOps are shorter development and release cycles [2], [5], [41], reducing projects' lead time [41] and making it faster to market [2], saving costs with a better ROI (Return on Investment) and improving reliability with continuous integration, since issues are found earlier, and general problems are discovered and solved [2], [41].

There are more, such as automation [2], [38], [39], [41], improved deployment management with predictable deployments because developers deploy their own code and discover flaws in it first [2], usage of various tools that can track the changes in several files, facilitating project management [5], [39], standardized process configurations [2], bigger responsiveness to change and shorter feedback loops [2], [41], in addition to greater trust and collaboration across all teams [2], [38], [41] that enhances their communication [38] and motivation [2]. The cultural change and infrastructure can play a role as advantages [38], moreover the information sharing, quality assurance and standards do too [41]. All of these ensure that DevOps improves software quality [5], [41].

Nonetheless, despite DevOps' advantages, there are challenges regarding its implementation, given the fact that there is usually resistance to change inside companies or organizations and [41] there is the need for a cultural change in project execution [2], [38], [39], [41] to properly implement DevOps [41].

Other challenges include the requirements for an extensive set of skills [2], [41] and for trust between the teams of development and operations, besides being difficult to implement [41], given the lack of general education and definition about the topic. When there is the DevOps implementation, the struggles are with software architecture [2], [41], because there are several applications and coding languages that influence deployment scripts [2] and assessment of the progress made, or the metrics used [41].

In the same way, business priorities and customers' needs are constantly changing [2], smaller development teams struggle more due to the tasks overlapping for them [2], [41], tight schedules and cooperation between multiple stakeholders [39] that can lead to insufficient communication [2], [41] are which contributes to DevOps' disadvantages. Even for project managers it is complex to monitor processes in various sprints, while performing continuous activities, like planning, integration and deployment [39].

Nonetheless, in 2021, 63% of the companies stated to have improved deployment and an acceleration in their release time, while 55% mentioned a better teams' collaboration with the adoption of DevOps tools [43].

2.6 Bottlenecks of Quality Assurance in Software Development

Around the years of 2019, the biggest part of the problems in software quality assurance were associated with human factors. Additionally, there was also some fear of possible gaps of people's knowledge due to the technological advancement [13] and the rising number of individuals with non-IT backgrounds [13], [44].

Even so, the obstacles faced during the software development process vary a lot, going from communication problems, like poorly defined initial requirements for the software [44] or bad communication among team members, to project management issues, such as lack of methodology, lack of best practices, mediocre management of stakeholders or insufficient support from higher layers, and even to compliance problems, for instance lack of auditing during

the software development life cycle, absence of a plan for maintenance, poorly executed risk assessments or even surpassing the planned costs [18].

Moreover, a considerable part of the difficulties in software development revolves around how to express requirements and respond to client feedback [13], but with recent evolution in processes it is also adopting a new methodology, while still having the mindset for the previous one, for example trying to implement Agile, but thinking the same as in Waterfall [45].

Another stated problem is the lack of adherence to the existing guidelines for software development. This happens because of two main reasons, being the first one the difficulty felt by workers for including these guidelines in their day to day and the second the fact that, for example in Agile, teams are independent and organize themselves, which can make it harder to coordinate the standardized processes and the commonly used practices. Many times, teams feel that these guidelines act as suggestions they can take into account and not as a procedure to follow thoroughly [45].

Furthermore, a study conducted in 2019 showed that some of the quality assurance challenges associated with software development were the reason for other ones. Additionally, introducing a specific practice could make some of these challenges disappear, while creating different ones [45].

The following chapters aim to explain more in detail some of the bottlenecks addressed in literature until now.

2.6.1 Code Reviews

The traditional code review was initially defined in 1976 by Michael E. Fagan under the term of software inspection. This was a highly structured process with the aim of discovering defects and relied on fixed roles for the participants and well-defined steps [46]. Accordingly, software inspections were included in the software development projects as part of the validation, since an effective inspection could detect and erase from 60% to 90% of the defects on the project. Some other benefits encountered were the improvement of team communication, the education of team members regarding avoiding the most regular defects or even the development of quality with the identification of flaws [19].

On the contrary, there were still some problems in 2010 with the application of software inspections: developers found it difficult and took some time to identify defects on their own

work, there was not enough guidance for people to know who to involve in the process and how to guide them and each person had their own ways to achieve the same steps [19].

Overall, inspections were perceived as a heavyweight process, not addressing the main problems related to the team [19] and being very time consuming, since they included meetings that required previous preparation by all the stakeholders [46].

These characteristics made software inspections a poor fit with Agile development methods [46]. As a consequence, in 2013, the "modern code review" appeared as a less heavy process [3], with the term formalized by Alberto Bacchelli and Christian Bird. This review, on the other hand, occurred frequently and was supported by tools, making it more suitable for recent methods [46].

Currently, the code review is an important action when it comes to software quality assurance [18], [46], since it is a critical activity for the code quality [3], [18]. This is achieved through the finding of bugs in the code [18], i.e. figuring out why the software is not behaving as expected and fixing the component causing the situation [47], through the verification of the desired guidelines, also contributing for the discovery of the best practices to code, through knowledge sharing [18], and spreading information about the project in general [46].

However, code reviews still include some problems, such as delays in making the necessary changes to the code, since reviewers struggle with the issue of understanding these changes (when they are not sure if they are valid, the impact they could have, etc.) and the context of them [46]. Additionally, the bigger the number of collaborators that is involved in a code change also increases the probability of defects that may come from it [14].

There are currently 30 different motives for confusion associated with code reviews and 14 different impacts this confusion can have. The 5 main reasons for the confusion are associated with extended and complex changes to code, lack of proper organization, lack of proper documentation, not understanding the reasoning behind code changes and lack of testing. The main ways confusion can impact code reviews are, as mentioned before, delaying the process, but also adding the necessity to have further discussions and even reducing the quality of the review [46].

2.6.2 Time Pressure

Time pressure can be referred to as "schedule pressure", "deadline pressure" or "time budget pressure" in the software engineering industry [16]. Given the fast changes that occur during the software development process, testers usually strive to keep up with developers [6]

and ensure Quality, but so do development teams [6], [19] that have it even harder, since it is common for them to be inside other departments [6].

Some causes that can lead to time pressure are the pressure that comes from customers, struggles with scheduling or estimating effort for certain tasks, someone's management style or the company culture. When the case is this last reason, then there is a probability of time pressure being a predominant factor in the company [16].

There can be a positive or a negative outcome coming from time pressure. The positive one is the sense of urgency that increases the team's efficiency. Even so, from the negative perspective, time pressure can generate tunnel vision, in which people start to value more short-term and fast solutions instead of long-term ones that would be more beneficial to the longevity of the product being developed, restricting the chances of improving the product or process and, consequently, decreasing the quality of the final product [16].

However, it has been found that increasing the pressure put on a team of developers can increase productivity, but only until a certain point. After reaching this point, there will be a fast decrease in performance. The negative effects for this and for other causes of stress can decrease when someone is given more independence and decision-making opportunities in their job, so managers and even software engineers need to be aware of the team's feelings about time pressure and the way it is being dealt with [16].

This factor appears to show itself mostly during the parts of quality assurance and testing, varying depending on the tasks' type, as tasks that possess a lot of numeric thinking have their efficiency and quality more harmed than other types of tasks [16].

Even so, the impact caused by it is different when the methodology applied is Agile or a traditional one. In Agile, there are small iterations that make time pressure little towards reaching the next deadline. Here, testing and quality assurance happen at the same time as development, which avoids that usual pressure on these two phases [16]. However, there is also the downside that whenever there is the factor of time pressure present, Agile teams tend to ignore the quality requirements and focus on delivering solely the functional requirements, since this is what the customer values the most [45]. On the other hand, in traditional development methods, there is a final deadline for the process without halfway deadlines, so the focus is all on the same date and this can lead to uneven workload during the process and piling the tension near to the end [16].

Even in the context of globally distributed teams, there is data to confirm that timelines and Quality are correlated positively, meaning that late projects tend to not achieve good levels of Quality. Usually, Quality ends up compromised so that teams can fulfil the deadlines [10].

The best way in general to achieve the positive side in time pressure is applying it in a little to medium level and avoiding the high one. Even so, it is advisable to apply time pressure in specific moments, increasing the efficiency of the process, but not cutting out buffer and reflection moments, to ensure a high-quality outcome [16].

2.6.3 Data Quality

In the beginning stages of a software development project, it is necessary to define the project's targets for certain metrics, for example defect density, and this is always done taking into consideration the resulting data coming from previous projects [5], [48]. Sets of data coming from finished software and its development process are used for all types of purposes, whether it is project management or quality assurance [48], since previous projects' data furnishes useful information on where is more appropriate to allocate certain resources in future projects [19].

There are several definitions and points of view for data quality's meaning, yet the one that is more accepted is being fit for a purpose, in the sense that data needs to be able to be interpreted and applied to a specific context to become useful and hence the purpose that appears in the proposed definition [48].

However, the main issue is that data sets' quality is not investigated before its usage, because there is not a big variety of metrics to quantify this data's fitness for a purpose. Unti 2022, CIL (Case Inconsistency Level) was the only proposed metric to assess data inconsistency in a data set and at that time there was a proposal for an improved one, the SCIL (Similar Case Inconsistency Level), that was empirically tested and proven to obtain elevated accuracy levels when evaluating consistency in data sets [48].

The overall bottlenecks associated with data quality can be split into three categories, being these trustworthiness, relevance (usefulness of the data) and accuracy, but the last one can even be broken in five sub-categories: incompleteness, inconsistency, redundancy, outliers and noise. Consequently, if there is poor data quality in past projects, then the estimation for the future ones will not be accurate and perhaps even damaging [48].

Even so, this is a very little addressed subject, since in 2022, there were only 23 articles mentioning this problem. As a result, this is a topic to which researchers should provide a bigger attention to [48].

2.6.4 Manual Testing

Either manual or automated testing tools can be used for testing purposes. Even so, with the growing complexity when it comes to developing software, teams feel even more the need to make use of advanced tools for testing to assess application's characteristics, such as functionality and quality [24].

Manual testing can be a long and arduous process that may not achieve a bug free code every time that is performed. However, it is a good choice for companies with lack of financial resources or for a tester that wants to verify almost all the features of the developed app [24]. Testing manually every day or week during the entire process of software development could be ineffective [6], since it takes more time than another option [8], [24].

On the other hand, automated testing tackles the bottlenecks that come with manual testing. Automated testing makes possible the creation of test scenarios that can be repeated and reused as much as necessary [24], so it is very useful whenever there is a big amount of test cases [6]. This type of testing is a more effective process whenever the variables of time, money and usability are considered [24], [49], so given these traits, automated testing improves testing's efficacy [24] and is one quality factor through which Quality can be ensured [13]. Another advantage for this is the identification of issues as early as possible in the SDLC, contributing to savings of costs and energy [8]. Because of all these advantages, knowing how to test automatically is a skill with higher demand than knowing how to test manually and this is in agreement with the rising usage of Agile methodologies [27].

However, some impediments for a quick implementation of automated testing tools are not only the high financial initial investment that is required, as it is not certain the return on investment that may come from this quality assurance measure [13], but also the need to train the people inside the company, which implies several types of resources [24].

A study in 2015 showed that most testers still realized testing manually, because they believed that updating the automatic tests took too much time, making this a difficulty in the industry due to people's persistence [27]. On top of that, according to a study in 2021, only 6% of professionals believed automated testing could be done as an exclusive practice, despite the benefits it could bring [13].

In conclusion, automated testing's goal is to boost testing, by confirming if the results obtained are the same as the expected ones, as well as if there are any incompatibilities and has been proven to do so in a much more effective way that manually [8].

2.6.5 Development Requirements

There are two types of quality related requirements: internal and external. The internal ones are related to maintenance, the ease for future modifications and the longevity of the system that are usually not easily seen at first sight but end up being noticed due to the increase of price for the system due to the increased maintenance. On the other hand, the external ones are quickly noticed by stakeholders and are related to specific functions, way of working and general performance that the system needs to have to be accepted. Research conducted in 2019 showed that everyone inside the companies that use this system was aware of this division of classification for requirements, but the internal requirements were only a focus whenever the system began to fail or when it was truly necessary [45].

In addition, the quality requirements are usually associated with functional requirements, which can lead to conflicts in priority allocation, for example in Scrum. This association can cause two equally important quality requirements having very different priorities allocated to them due to the functional requirements, therefore causing only one of them to receive the deserved attention and the other one being ignored or even eliminated from the backlog [45].

The bottlenecks associated with development requirements are of different natures and can go from vague requirements from the customers' perspective [18], the requirements asked by the client not appearing in the final product [18], the way that requirements are expressed by clients [13] and the fact that requirements can be interpreted from many ways and there can be a tendency to focus only in one point of view and disregard the others [45].

Additionally, communication with the customer can be a risk due to three motives. The first is that the set of people responsible for designing and developing the software can be different from the ones that make the decisions. This can result in future problems inside the projects, due to the variation that may occur to the initial list of requirements that is not properly verified and approved by the ones in charge. Then, the second reason is the constant change of mind coming from the developers about the requirements and not keeping registered track of it, leading to struggles on the contracting responsible people to understand and accept the changes. Lastly, the fact that there is a lot of misalignments between customers and developers due to each side misunderstanding or making assumptions about the other, so it becomes very important that there is evidence to support the changes made, otherwise all the faults will always fall on the developers' side. All of these contribute for the high importance of keeping record of the requirements and of the person that asked for them, as well as the proof of acceptance [1].

On the other hand, the bottleneck of not properly establishing specific and long-lasting determined requirements for the software to be developed leads to a growing litigation associated with software creation projects. Therefore, not fulfilling the initial requirements or not establishing specific and long-lasting ones is seen as misconduct and negligence, which can be interpreted as not fulfilling the contract obligations. Nevertheless, requirements evolve and can be unpredictable at certain times, making it very difficult and undesired to have definite ones, especially in very long projects. To avoid legal complications, it is important that organizations have an ongoing and transparent communication with their stakeholders and that everyone accepts to change the predefined requirements [1].

2.6.6 Teams Working in Different Locations

Currently, software development teams distributed across the globe are a regular way of working. Despite this being a normal reality, this trait can aggravate the already known issues in the industry [10].

A study conducted in 2013 focused on studying some factors, such as the effect that time zones could have, plus the impact on productivity and quality levels. It was stated that there was no correlation between the type of methodology implemented and the outcome of software development in this type of teams. Therefore, it would need to be other factors influencing globally distributed teams' success and not the fact the approach used was a traditional or an Agile one. The existence of strong and cohesive teams with moments of team building was determined a factor that increased team members' performance. The skills of the people involved in the project were also an important aspect to determine the project's outcome. There was data to confirm that timelines and Quality were correlated positively, meaning that late projects did not tend to achieve good levels of Quality and that usually, Quality ended up compromised so that teams could fulfil the deadlines [10].

Two of the explanations behind a positive outcome were if the company is comfortable and used to a wider range of customers, besides working for a short-term goal rather than a long-term one. These characteristics make these companies improve their skills, by being familiarized with different ways of working, and demonstrate a flexibility mindset that allows them to react and fix faster the bottlenecks that may appear [10].

Some of the challenges that can be faced by globally distributed teams are related to their communication, for example understanding too late that a certain requirement is not possible to be fulfilled due to lack of discussion about it, not specifying the assumptions being made by some to complete tasks or to follow another direction, having different maturity levels

for people inside the team and being disorganized [45]. However, in the previously mentioned study, one of the main points of success in globally distributed teams was communication, since most teams mentioned having a weekly meeting with everyone involved in the project, and no problems were reported regarding the different time zones and their effect on communication [10].

Other major challenges investigated and confirmed to be present in this way of working were lack of ongoing interaction between developers and customers and the lack of customer involvement, which led to a serious of doubts about the features to be included and overall, the problems described in the previous chapter. Testing quality requirements, inappropriate requirements' prioritization, lack of customers' domain on the subject, difficulties in documentation and involvement of all the necessary stakeholders in this context were also considered as challenges, proving the previous point of independence regarding the methodology and the success of this type of process [45].

3

METHODOLOGY

Firstly, to assess what was already being studied, thorough research on several topics was conducted. This was meant not only to build this dissertation's literature review, but also to understand what main quality related bottlenecks were identified so far in software development. After gathering everything, then the information could be compared with the information collected in previous related studies on the topic investigated empirically through some interviews as well as a final questionnaire.

Given the fact that there was a final questionnaire, the methodology used to fulfil the purpose of the current study was a survey methodology. This would allow the study to receive real feedback and inputs from employees of IT companies and IT departments, thus confirming the research gathered from available articles so far and adding up to the previous studies.

To execute a proper gather of information, Scopus was used as the main indexing base, through which several research was executed with different strings, in order to obtain as much variety of information possible, covering all the important topics related to the current study. The way the research was conducted can be briefly analysed through Figure 1.

It is, however, worth mentioning that due to the lack of open-source information available in the indexation bases about certain topics, since this is a field in which some research gaps exist, sometimes it was necessary to resort to reviews and conference papers, otherwise it would not be possible to address these subjects.

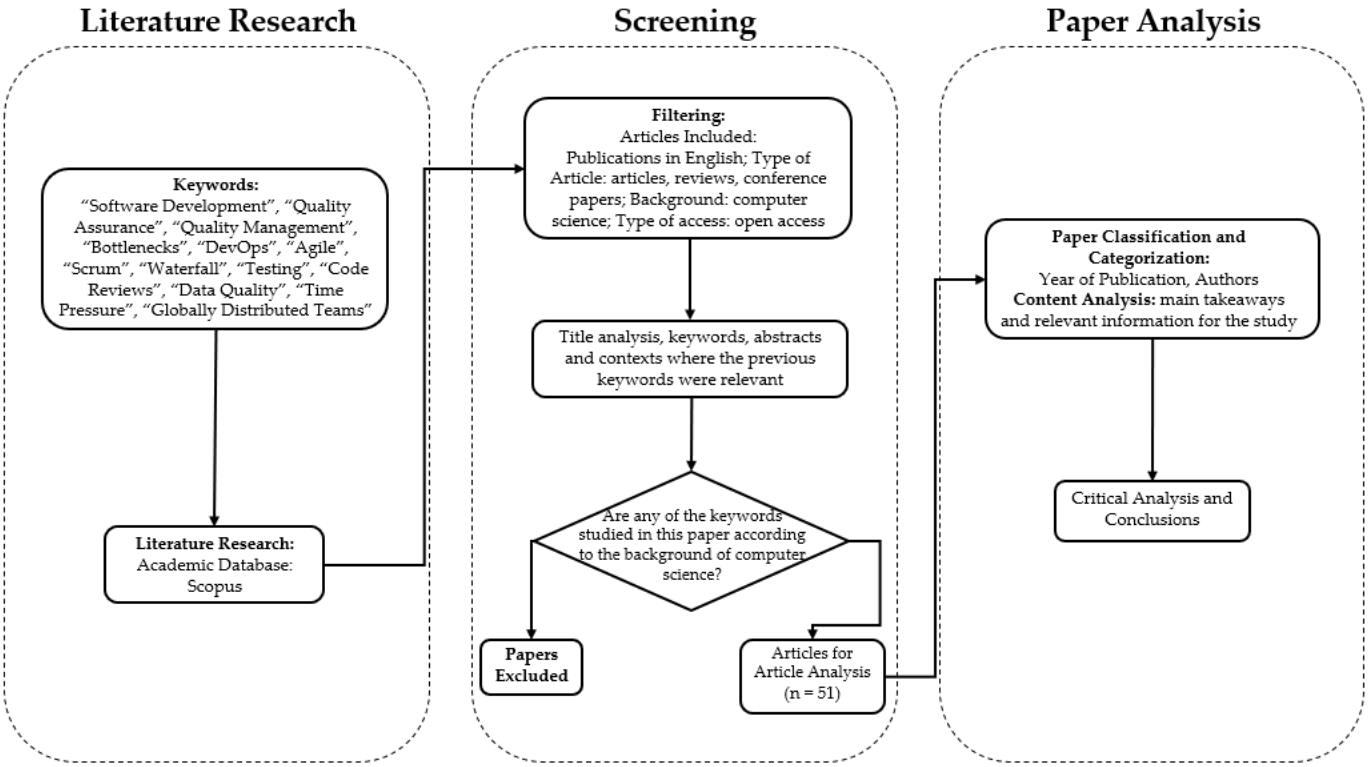

Figure 1 - Flow diagram of literature research and screening

3.1 Interviews

Four interviews were conducted to obtain some data to compare with the literature review and the previous information from this study to create a questionnaire that would be later on released to the general public.

These interviews were conducted to people that worked in IT companies or IT departments, with functions that could be related to quality assurance in the field of software development. In the specific case that this was not possible, there was a focus to collect the maximum amount of information on the company's processes and way of working, to have the information as accurate and relevant as possible for the current study.

Before starting any of the interviews, it was required to collect some general information about the interviewed people and then the company they work in, always with the guarantee that no identifying information would be disclosed. From the respondents, it was important to gather their age, position inside the company and professional background/general qualifications. On the other hand, from the companies it was relevant to have knowledge not only about their age, but also about the number of employees, its location, core activity

and if the answers provided were related only to an IT Department inside the company or if the company itself was IT oriented.

The script for these interviews was divided in three parts: quality perception and functioning, challenges faced in quality management and quality culture.

Part I – Quality Perception and Functioning

The questions for this part started by understanding what Quality meant for the company, since "it means different things to different people" (24), as stated before. After this understanding, it was necessary to assess quality assurance and management inside the company, mainly the company's structural organization (for example, if there were teams or it was a single person responsible for everything), the processes, methods and tools put into place to ensure quality and finally the metrics used by the company to evaluate its performance in this area. These questions facilitated the performance of a more in-depth analysis of the answers, since it allowed to establish a baseline of comparison between the different answers.

1. What does Quality mean for the company? When do you consider whether there is Quality or not?
2. How is Quality managed inside the company?
    a. If not mentioned: is there a person or a team(s) responsible for it?
        i. If there are teams: how is the distribution of duties between these teams/units?
    b. If not mentioned: which processes/models/tools are put in place to ensure there is Quality management?
3. Which metrics is the company assessing to evaluate Quality performance?

Part II – Challenges Faced in Quality Management

This part of the questionnaire is the key goal for this study, so the only main question is to acknowledge the specific problems that companies have towards achieving maximum Quality when developing software. However, with the possibility of the interviewees forgetting to cover some topics that could be useful when designing the final questionnaire, there were several follow up questions to confirm if the topics read about in articles and previous interviews truly happened in practice and were still up to date, given the fact that some were written a couple of years ago.

1. Which factors may be preventing the company from having (an even) better quality assurance established?
    a. If not mentioned: are there any difficulties regarding the lack of detail in requirements made by the clients?
        i. If there are: which consequences does the company face with the issue?
            1. Which procedures are being used to manage the requirements?
    b. If not mentioned: how is the coordination between developers and testers and how is it managed? Are there any difficulties in terms of code reviews, if used as a methodology?
    c. If not mentioned: how do quality assurance processes work? Are they tailored to each project, or do they have a strict guideline to be followed?
    d. If not mentioned: is data quality also a factor? (it may influence other aspects, such as planning of metrics in future projects)
        i. If it is: what is the main difficulty in ensuring data quality? (maybe defining quality standards, being too complex to measure it or even testing and validating it?)
        ii. If the previous points are mentioned: which procedures are being used to manage the standards, testing and validation?
    e. If not mentioned: how is the balance between adaptability and flexibility with process definition and standardization?

Having in mind that there is more flexibility in processes and there is usually not a standardized process to ensure Quality in the IT industry, it was important to comprehend the existence or not of a standardized process for Quality assurance.

Part III – Quality Culture

To understand the discrepancies that could lead to the problems assessed previously, it was necessary to ask about the organizational culture of the companies, specifically related to Quality. This way, it would be possible not only to evaluate if the companies with the biggest number of bottlenecks were the ones with a lack of a Quality culture and all the variations associated to this, but also if this topic was a general concern for the companies.

1. Is there a Quality Culture inside the company? Why?
    a. If there is: what is it like?

b. If there is: does this culture vary from one department to the other?
c. If not mentioned: is it common for employees to discuss the best practices together for their tasks or is there more a focus on individual talent?
d. If not mentioned: what is the leadership's vision when it comes to this "quality culture"?
e. If not mentioned: what resources are being allocated to Quality in the company?

With the script presented, four different people from different companies were interviewed to understand if the previous studies and the problems gathered during the literature review phase were realistic.

3.1.1 Company A

Company A is a consulting company that provides software quality services to other companies, based in Porto, despite having other locations, and with around a thousand people. The person interviewed from this company is 35 years old, with a bachelor's in computer science, a master's degree in computer science engineering and 5 years in the area of software quality, being currently the team leader of automation in a specific project.

According to the respondent's perspective, the company states that there is Quality whenever the developed software matches the one wanted by the customer, based on the written documents with all the requirements. Despite the proper definition for Quality, there are no fixed metrics to assess it, because they depend on the project and the type of testing.

The organization consists of several projects to which different people are allocated and the people of Quality are no exception, so there is a general team leader for the project team and then there are people for different ends, like automation or functional testing.

The developers' team is subcontracted in India and for this reason the main tools used by the ones in Portugal are Azure and DevOps, to follow up on the existing bugs. Their functions are essentially testing the components in three different environments and, once they pass all these, then there is a check to see if all the functionalities are working all together as planned. The entire process towards having a new release usually takes up to a month and there are not metrics to assess if things went right or wrong.

In terms of struggles, the main one is the different location for developers and testers, in the sense that there are communication misalignments at times. The initial requirements' book that is created is sent to both development and testing teams, but it has happened that developers had a meeting with the customer and some things changed, but they did not warn the

testers, so these last ones were creating testing scenarios for functionalities that were no longer needed. Other present bottlenecks are the lack of specification in some requirements that lead to confusion among the team and the absence of code reviews.

On the other hand, the interviewee mentioned there is focus on education about Quality from the company's side towards its employees, since there are ongoing sponsored trainings (two per month, at least) that can go from English classes to a software quality certification.

3.1.2 Company B

Company B is an IT consulting company with a focus in telecommunications that can develop software from scratch, give support in the creation of software together with the client company or improve the software already created by them. It is more than 20 years old, having been created between 2000 and 2001, with its headquarters in Lisbon, but with other locations such as in Ireland and Abu Dhabi. The interviewed person is 27 years old and is a functional tester with two years and a half of experience in the area since the end of a master's degree in quality management.

For this company, Quality is obtained when the team fulfils the definition of done, which varies according to the project, and when the customer's expectations are met with the software having the highest level of usability possible, preventing future correction costs.

The Quality and analysis department is the main responsible for Quality management, so there is a distribution of all the people involved in it through the areas of functional testing, analysis and automation, training and quality practices. The entire department is distributed by the active projects, but everyone brainstorms about improvement.

The implemented methodologies are Waterfall, Scrum and DevOps. In terms of general processes, the concepts of definition of ready (if the features to be implemented are ready, so all the testing and requirements are defined) and definition of done (the developers complete the software and, to deliver the new feature, all the requirements are analysed) need to be well defined at all times. Regarding metrics, defects' density is the one used the most for their projects, being measured in several testing environments, depending on the project. In the respondent's project there are three testing environments: the internal environment in which the first bugs are detected right away, the client's testing environment in which it is not supposed to find a lot of bugs and then the real environment in which it gets more serious if any mistake is found, so these should be zero. Nonetheless, general KPIs differ according to the project.

In the specific project where this tester is currently inserted there are not many problems, but the main bottlenecks identified in general inside the company were priorities' management and lack of resources, as well as the fact that each project follows different rules regarding Quality, even if all the project's guidelines are followed. Some of the practices that lead to this alignment and lack of major issues with the project are calibration meetings with the client to facilitate communication about requirements and ensure the project is going in the desired direction, alignment meetings between testers and developers and training sessions about best practices for the employees.

There is a culture of quality inside the company as the sub department responsible for best practices makes sure there is a dissemination of this culture, making Quality an everyone's responsibility and not only their department's. The initiatives that promote this culture are inviting people to give sessions to different teams on their best practices, the amount of communication channels inside the company and the ongoing transition of deliverables to become automatic instead of strictly manual. Inside every project there is also a lot of reporting and the confirmation if the metrics are according to the initial planning, adding to meetings with everyone to discuss the scenarios considered for testing.

In the last years, there has been a growing concern about Quality on the company's behalf, so there is a Quality responsible team in every project the company makes and the department itself has been hiring more people.

3.1.3 Company C

Company C is a 6-year-old software development company that only produces software for a specific company. It is based in Porto, even though it has offices in Lisbon, which totals an amount between 2500 and 3000 employees. The interviewee is 35 years old and originally had a background in multimedia, but after being a freelance developer, became a postgraduate in Computer Science, so currently develops software and works with DevOps.

There is no Quality department, so the quality tests are performed by the company for which this produces software or by others. However, each team is responsible for their quality management, so whenever a new software component is created, it needs to be tested right away first individually and then its integration.

Additionally, inside every team there is a validation criterion. Since the methodology used is Scrum, there is also a big importance attached to the definition of ready and definition of done (the definition for both is the same as in Company B).

This company uses some tools for automated testing that indicate the level of compliance with the company's main quality gates of the tested code.

The main issues stopping the company from achieving an even better quality assurance are the lack of requirements' definition from the clients' side and the very long thread of people through which the information passes until it gets to the developers. This leads to difficulties in communication and alignment, especially because it makes the developers pressured to work towards something that sometimes turns out to not be what was really wanted.

In addition to these, time pressure is often a bottleneck, due to the fact it makes people forget Quality and just focus on delivering the task despite the result. Nonetheless, this results in code with lack of Quality that it is more complex to fix after.

Other problems stated were, firstly, the high independence of all the developers, because not everyone enjoys the lack of guidelines and supervision coming from higher layers, and second, the lack of a standard procedure that everyone can follow, which makes people use their own experience to create the code, so the results can differ a lot from one person to another.

It is spread in the company an Agile culture of excellence in engineering with a focus on results' transparency and there are several initiatives that contribute to it, such as a culture of feedback inside and between teams, an individual process of personal development for all the employees, knowledge sessions with different useful topics for everyone and a conference day with workshops given by people from the company.

3.1.4 Company D

Company D is a financial software company created in 2015, in the United Kingdom, and currently has around 8000 employees. The interviewee is 32 years old, initially with a bachelor's degree in textile engineering and then an incomplete master's degree in quality management, currently performing the role of quality analyst in the complaints' area of the company.

This interview is a particular case, because the area from which the respondent is part of is not inserted in the quality wing of the company. Nonetheless, there are three specific quality teams, one of them for complaints' evaluation, another for analysts' coaching and the last one for customer support through chats. As a general comment, the respondent believed that Quality is ensured with a broader approach.

There were no specific methodologies mentioned, but the main processes behind their work were assessing the root causes for the issues encountered, analysing customers' feedback and refreshing the guidelines for the procedures inside the company. The interviewee also

stated that even though there were KPIs to track Quality, sometimes they were very subjective and qualitative instead of quantitative.

As main problems covered, these were the spread of information inside the company, the lack of clarity regarding procedures and data quality. When it comes to the communication, this happens particularly because the developed application is in constant change, which makes it difficult for everyone to always follow along and understand the features that were already released, generating errors daily. Despite this constant change of the application, the same does not happen to the procedures that define people's work around it, so it takes a while until workers fully understand the adaptations to their job with the new app features and there are even contradictions on the available information for ways of working. Last but not least, data quality is an issue, since metrics are defined subjectively and new frameworks are created based on these types of metrics, sometimes ignoring the fact that these metrics are not appropriate for the new framework, which complicates processes' tracking.

An additional bottleneck related to the previous ones mentioned was the lack of automated processes and thus the human errors being a constant for this organization.

Despite the lack of structure, the interviewed person believes there is a culture of quality in the sense that everyone feels comfortable to ask for help and discuss the best practices, there is a lot of work done as a group, the company's goal is very directed to customer satisfaction and the mindset installed is an overachievement and continuous improvement one.

3.2 Questionnaire

Once the interviews were completed, the information gathered with the output of the literature review allowed the construction of a Google Forms questionnaire that would be the main research item to be spread for as many companies and IT/Quality oriented workers all over the world.

The main difference from the interviews to the Google Forms were the predominance of multiple-choice options, while in the interview all the questions had an open answer, with only some questions leaving space to add more details if needed. This strategy usually accelerates people’s responses to questionnaires and was also useful for the analysis when filtering similar information that was always written in the same way because this format was used.

To reach the desired public, given the restricted audience for this study, several software development companies were contacted via email and multiple Quality/IT related

LinkedIn groups received a post about the topic. While these two strategies were implemented, there was a spread of the word among known people.

The released Google Forms included 24 questions, the response time varied between 5 to 7 minutes, and it was divided in five sections: Respondent Profile, Company Description, Understanding Quality, Quality Culture and Quality in IT.

Before jumping into the previously mentioned sections, it was important to have a statement from each participant declaring that they consented in giving away this information for the present study to be conducted. Therefore, the first question of the survey was "Do you allow the answers to this form to be analysed for research purposes?" with the options of "Yes" and "No", so that in case there was any "No", the survey would finish right away.

After that, the remaining questions appeared and some of them were only available according to the answer to the previous question, so the data could be as accurate as possible. The full questionnaire is available in Appendix A.

Each question of the questionnaire also had its purpose, so that each section could make it possible to draw a conclusion or correlation with another topic:

- Section I - Respondent Profile - Five questions entirely about the person answering the survey, because according to the age and number of years in the industry, the role inside the company would change and probably influence the way the topic of Quality is seen, as well as the main struggles of SQA.
- Section II - Company Description - Similar to the previous section, the company description questions were asked for a possible comparative analysis of Quality's organizational structure between companies situated in different continents, with a different range of number of workers and with different market positionings.
- Section III - Understanding Quality - Focused on the perception of Quality for each company and on the way, it is managed in terms of organizational structure, methodologies and metrics. This would allow a better understanding of the companies' perspective on when they would be achieving Quality and the way to do so.
- Section IV - Quality Culture - This section was an assessment on the type of initiatives promoted by companies to foster a culture that promotes and enhances Quality in their processes. Therefore, it was important to have the respondents' perspective on whether there is a culture of quality, as well as how it was created and spread, specifically if the employees felt that the implemented initiatives were building and fostering this culture.
- Section V - Quality in IT - Consisted of three questions, being the first two ones the collection of bottlenecks that could be an issue for the company on their pursuit for Quality excellence and then the elements that best define Quality specifically related to the IT industry. To conclude the survey, and to establish a bridge towards the future of

the sector, there was a question to assess how each component of software development would be impacted by Artificial Intelligence.

4

RESULTS

Through the divulgation of the created survey, 50 answers were gathered and analysed through Google Forms and Microsoft Excel. The following sections contain the analysis of their respective questions, even if sometimes necessary associations across sections are made to prove the data is coherent or to trace some conclusions for the study.

4.1 Results' Analysis

Section I - Respondent Profile

From the 50 respondents, 38 claimed to be between the ages of 18 and 45, which represents 76% of the sample collected. Even excluding the people between the ages of 36 and 45, still left an amount of 26 people that is correspondent to 52% of the sample. These numbers revealed that the answers collected were from a general young public.

In terms of qualifications, 30% of the people indicated they have a bachelor's degree and at least 56% have a master's degree, existing even a 10% amount that is in possession of an MBA, so the respondents can be considered qualified.

The bigger portion of the respondents have been for a short period of time inside their current companies, with 60% being there for less than 5 years. The next 30% has been between 5 and 20 years, so the current data appears to be consistent with the range of ages previously analysed, since people around the ages of 26 to 35 are not expected to not have had very long careers so far.

Also, in line with the previous answers about their age, 48% of the respondents have less than 5 years of experience, with the 52% remaining being distributed among the other options, which creates a good balance between people with and without long work experience.

Most respondents were developers, with a 36% presence in the current sample. Other roles like quality assurance engineers have a relevance of 12% and then quality managers and testers constitute 8% each. Considering the ages and work experience mentioned previously, it makes sense that the percentage of answers for higher organizational layers, such as business heads or CEOs is significantly less than the one for lower layers, more connected to the development of the software.

Section II - Company Description

The vast majority of the answers to this survey belong to people whose company has a software related purpose and only a small group of people answered about the IT Department they are part of, since their company is not focused on IT specifically (Figure 2).

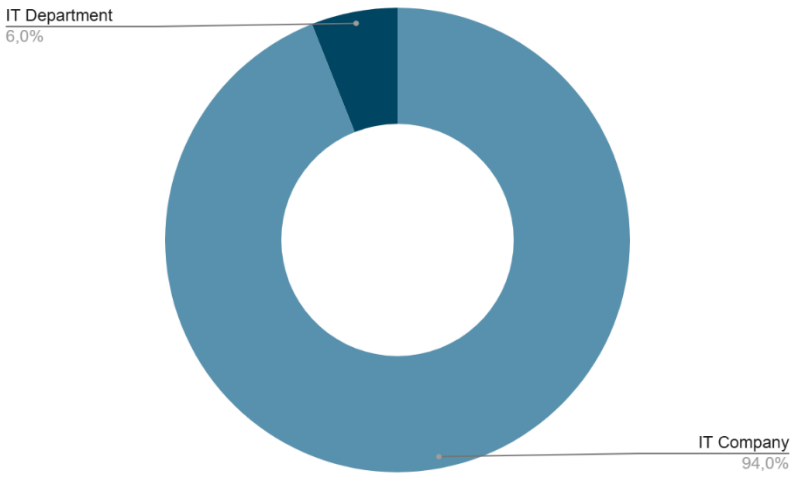

Figure 1 - Distribution of answers to the question "Is the core business of your company IT or are you part of an IT department supporting operations?"

In agreement with the previous answer, the core activity most pointed out was software development, with a majority of 64%, while 18% claimed their focus was selling and managing a software platform, 10% were focused on IT Consulting and 8% gave other type of answers.

In terms of the companies' growth, the biggest part is considered an established player even if they are not a national, regional or global leader, represented by 40% of the results. The second highest score was for regional/global leaders, with 26% of the answers and the rest had less significant percentages.

The percentages that represent the companies' size are shown in Figure 2. However, it is worth noticing that 70% of the companies had between 250 and more than 2500 employees, so the sample collected was within a range of generally big companies.

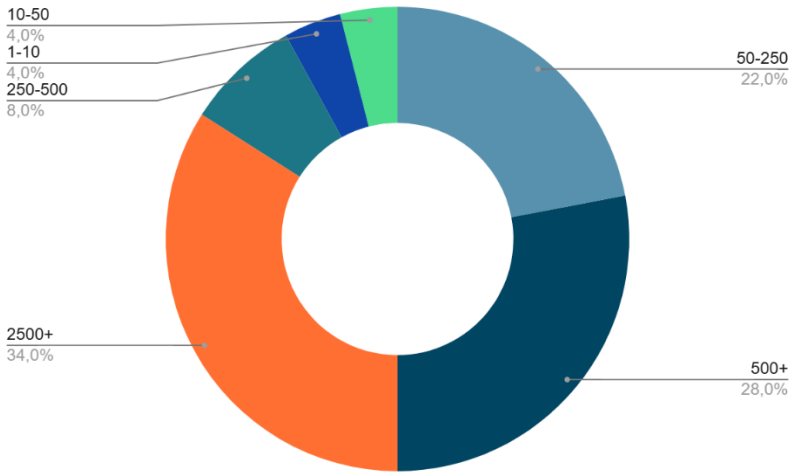

Figure 2 - Distribution of the answers to the question "What is the number of employees inside the company (at your site)?"

Furthermore, the high number of employees as top answer is aligned with the majority of the companies being already at least established players and not startups, that traditionally are constituted by a smaller group of people.

Unfortunately, it was not possible to collect any data from America and 76% of the answers came from European companies. In addition, the percentages for Asia, Africa and Middle East were not significative, but despite the lack of representation for these three regions, the answers collected for Asia and Middle East were all from companies with 2500 employees or more, while the ones collected for Africa went the opposite direction, being from companies with a number of employees between 10 and 50.

Section III - Understanding Quality

When asked about their understanding of Quality within the company, people mostly mentioned ISO Standards and customers' requirements fulfilment (Table 1). Even with a small percentage of difference and being inside such a specific industry where Quality could be associated with a lot of different topics, it was an interesting revelation that the majority still associated ISO Standards with Quality.

Table 1 - Distribution of answers to the question "What would best describe the understanding of Quality within your company?"

Topics associated with Quality	Total	%
ISO Standards	16	34%
Specific project metrics	8	17%
Customer's requirements fulfilment	11	23%
Stakeholders' satisfaction	9	19%
Density of errors/defects	2	4%
Specific metrics established by the Quality team	1	2%
All of the above	3	6%
Total	47	100%

The entity mainly responsible for Quality inside the companies of this sample is a Quality department, with 46% of the votes (Table 2). Only companies with, at least, 50 employees have Quality departments, yet 78.26%, i.e. 18 out of 23, of the companies with a Quality department have a minimum of 500 employees. On the other hand, there are only 4 companies with more than 250 employees that have between 1 and 4 people controlling Quality, which means that 73.33%, i.e. 11 out of 15, of the companies with less resources allocated to Quality have a maximum of 250 employee. Overall, the answers to this question were coherent with the answers to the question about the number of employees and the general rule is that the bigger the company, the bigger the number of human resources allocated to Quality and the creation of specific departments for it.

Table 2 - Distribution of answers to the question "Who is responsible for Quality inside the company?"

People responsible for Quality	Total	%
One single person	5	10%
Small team (2 – 4)	10	20%
Large team (5 – 8)	10	20%
Quality task force (8+)	2	4%
Quality department	23	46%
Total	50	100%

More than half of the answers (54%, to be precise) on the methodology implemented to develop projects were "Scrum". After that, 16% mentioned that they implemented some or all the methodologies present in this question, meaning that they have a hybrid system, and after that 10% claimed to implement DevOps. There were no answers that mentioned the

exclusive implementation of the Waterfall method and other types of Agile methodologies, besides the very few answers for XP (type of Agile methodology) and other type of traditional methodologies.

The most used type of metrics are project related ones, yet the metrics regularly used do not vary a lot according to the companies, with a difference of only 2% and 4% from customer satisfaction to project and project to product, respectively. This made customer satisfaction metrics the highest ranked ones, with 36%, project related the second ones, with 34%, and lastly product related ones, with 30%. Regarding project metrics, the most used one is time, then cost and delay with the same value, as the more significative ones. When it comes to customer satisfaction related metrics, percentage of compliance with customers' requirements is used by almost a half of the companies in this category, followed by NPS. Finally, in terms of product related metrics, defects' density and number of vulnerabilities are the most used, with 28% of the answers, followed by the number of code smells (a code that is smelling holds quality concerns that can lead to bugs in later phases or restrain the code's maintenance [50]), with 24%, and the remaining 20% for other briefly mentioned options.

Table 3 - Global weight of the implemented quality metrics

Metrics	%	Weight	Total
Time	35,6%	34%	12,1%
Cost	28,9%		9,8%
Delay	28,9%		9,8%
Other	6,7%		2,3%
NPS	40,7%	36%	14,7%
%Compliance	48,1%		17,3%
Other	11,1%		4,0%
#Code smells	24,0%	30%	7,2%
#Vulnerabilities	28,0%		8,4%
Defects Density	28,0%		8,4%
Other	20,0%		6,0%
Total		100%	100%

Having in mind the percentages obtained for each type of metrics and considering it a ponderation regarding the whole set of metrics, then it was possible to understand the most important metrics in general for the companies. With this, and according to Table 3, 17.3% was the first place for %compliance with customer requirements, 14.7% was the second place for

NPS and 12.1% was the third place for time. There was a tie between cost and delay with 9.8% and between number of vulnerabilities and defects' density with 8.4%. The remaining metrics did not have a significant percentage.

Section IV - Quality Culture

Most of the respondents claimed to have a culture of quality inside their company (Figure 4).

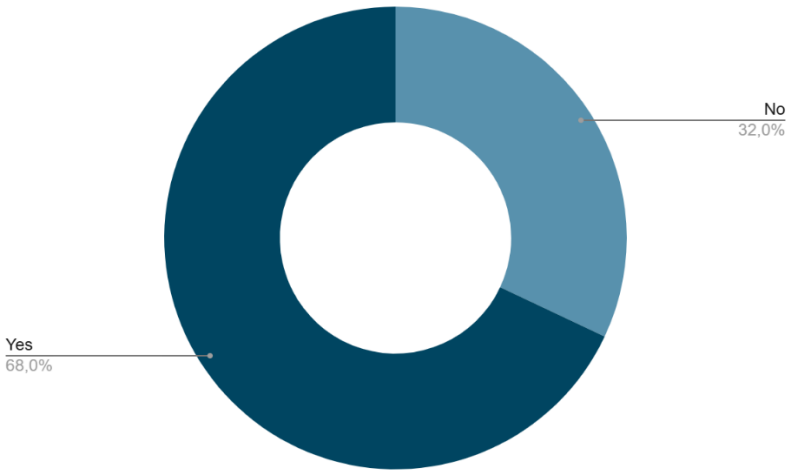

Figure 3 - Distribution of answers to the question "Would you say there is a culture of Quality in your company?"

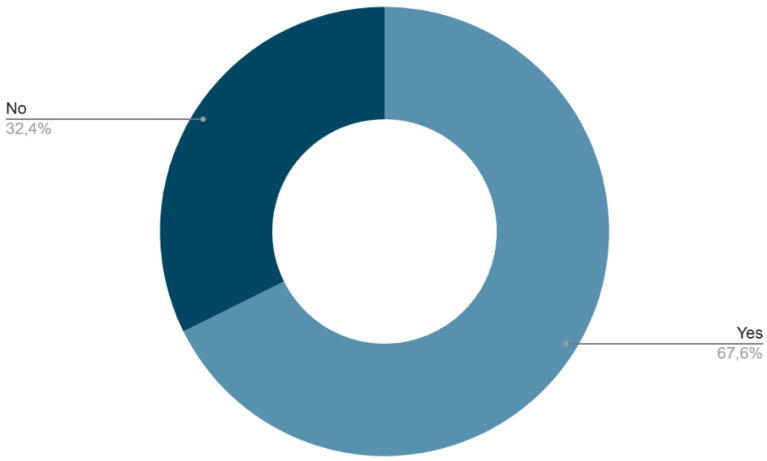

Figure 4 - Distribution of answers to the question "Does this culture of Quality vary according to the department of the company?"

Of the 68% that answered "Yes" on the previous question, 67.6% mentioned that even though there was a culture of quality, it was different according to the department (Figure 5), which may represent a creation of microcultures inside the company that, without sharing good case practices or tools, consequently, create more resistance to the ongoing improvement.

Answering about the characterization of Quality inside the company, and according to the Table 4, 33% identifies the development of dedicated metrics, tools or models to manage Quality as the more important factor. Even so, "outspoken focus on Quality across corporate communication" and "Quality Awards or recognition from customers & stakeholders" were the next two main factors contributing for this culture of quality inside the companies. There was not a lot of diversification on the characteristics implemented by the companies.

Table 4 - Distribution of answers to the question "What better characterizes this culture of Quality?"

Quality characterization inside the company	Total	%
Outspoken focus on Quality across corporate communication	20	25%
Development of dedicated metrics, tools or models to manage Quality	26	33%
Quality Awards or recognition from customers & stakeholders	19	24%
Training	13	16%
Other	1	1%
Total	79	100%

The majority reported having regular discussions with colleagues about the best practices to execute their tasks (Figure 6), which is coherent with the fact that the majority also identifies there is a quality culture inside the company. Nonetheless, there is a bigger percentage for "Yes" on this question (76%) rather than on the previously mentioned one (68% on Figure 4), which can demonstrate one of two things: 1) people do not think this sharing is enough to say there is a culture of quality or 2) they do not know this can be one of the practices that indicates the existence of a culture of quality inside the company.

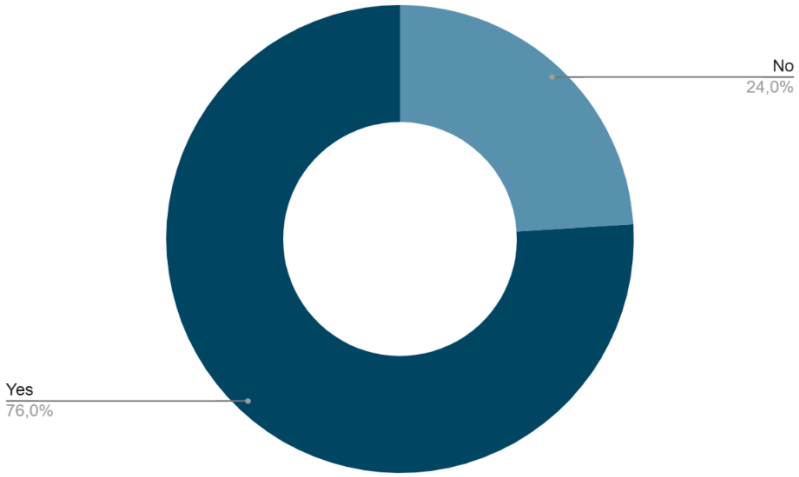

Figure 5 - Distribution of answers to the question "Is it common for employees to discuss the best practices together for their tasks?"

To spread a possible culture of quality, the biggest part (32%) shares the best practices among different teams, so that everyone can improve their work despite the area. Some companies (20%) have implemented a feedback culture through specific platforms and practices, while others have certified quality related courses (8%), ongoing workshops related to Quality (8%) at their disposal or individual development plans (6%). However, the third most mentioned answer was "None", with 12% of the answers.

Moreover, something worth noticing is that, from the 16 people that answered "No" when asked if there was a culture of quality inside their company, only 6 were part of the group that did not identify any initiative implemented. This demonstrates that some companies are not including Quality as one of the topics on their agenda and that others have employees that are not recognizing these adopted initiatives as enough to foster the desired culture.

Section V - Quality in IT

Question 23 had the purpose of understanding which three elements from the software development process were more related to Quality, in people's perspective. Counting all the answers, there was a tie between methodologies and organizational culture, so since there were 8 elements, it was assigned a multiplier from 1 to 8 to each element. The element with the highest total had the highest multiplier and the lowest total had the lowest multiplier, but in the case of the two elements with the same total, the multiplier that would be 5 for each was converted to 4.5. This made it possible to create a ranking of all the elements, as presented in Table 5.

Table 5 - Software development elements' ranking according to the answers to the question "Choose the 3 elements that are more specific regarding quality in IT."

Elements	Total	Multiplier	Rank
Requirements	35	8	1
Development	15	3	6
Testing	30	7	2
Processes	19	6	3
Methodologies	17	4,5	4
Tools	10	2	7
Organization culture	17	4,5	4
Organizational structure	7	1	8

After this analysis, the number of votes for each factor presented as a possible reason for failures in quality assurance was counted and every factor was associated with a requirement, so for example, "poorly defined customers' requirements" was linked to the element of "requirements". Based on the assigned element, each total was multiplied by the previously defined multipliers and considering the total of the column "xMultiplier", the percentages of each element were calculated, as demonstrated by the Table 6.

Then, since the differences in the percentages were not allowing to clearly understand which factors were determining the most the success of software quality assurance, the factors that were part of the same element had their total from the column "xMultiplier" summed to create a total per element and, consequently, a percentage of impact for each based on the total of that column. So, for example, the element "Requirements" was a sum of 144 and 96, since the first two factors of the Table 6 are included in this element. The total results are displayed in Table 7.

Table 6 - Association between the factors that prevent SQA and software development elements

Elements	Factors	Total	xMultiplier	%
Requirements	Poorly defined customers' requirements	18	144	14,0%
Requirements	Communication with the customer through the development process	12	96	9,4%
Development	Priorities management	26	78	7,6%
Testing	Code reviews could improve	12	84	8,2%
Organizational Culture	Independence of the team members	11	50	4,8%
Testing	Definition of quality standards	19	133	13,0%
Testing	Manual testing	8	56	5,5%
Development	Time pressure	26	78	7,6%
Processes	Data quality from previous projects damages the planning of new ones	7	42	4,1%
Organizational Culture	Ignoring the best practices	10	45	4,4%
Methodologies	Lack of coordination between developers and testers	9	41	3,9%
Processes	Quality assurance procedures not adapted to software development projects	13	78	7,6%
Organizational Structure	Limited resources	13	13	1,3%
Testing	Planning the testing and validation phases	2	14	1,4%
Organization Culture	Lack of a culture of quality	16	72	7,0%
Organizational Structure	Teams working in very distant locations	3	3	0,3%
Total		205	1026	100%

Table 7 - Relative frequency of each software development element

Elements	Individual cause	Accumulated cause	Relative Frequency	Accumulated Relative Frequency
Testing	12,5%	12,5%	28,0%	28,0%
Org. Culture	12,5%	25,0%	23,8%	51,8%
Requirements	12,5%	37,5%	23,4%	75,2%
Processes	12,5%	50,0%	11,7%	86,9%
Development	12,5%	62,5%	7,6%	94,5%
Methodologies	12,5%	75,0%	3,9%	98,4%
Org. Structure	12,5%	87,5%	1,6%	100,0%
Tools	12,5%	100,0%	0,0%	100,0%
Total	100,0%	-	100,0%	-

Finally, it was possible to have a total for each element that made it possible to understand the ones impacting more software quality assurance's success and a Pareto Diagram was created (Figure 7).

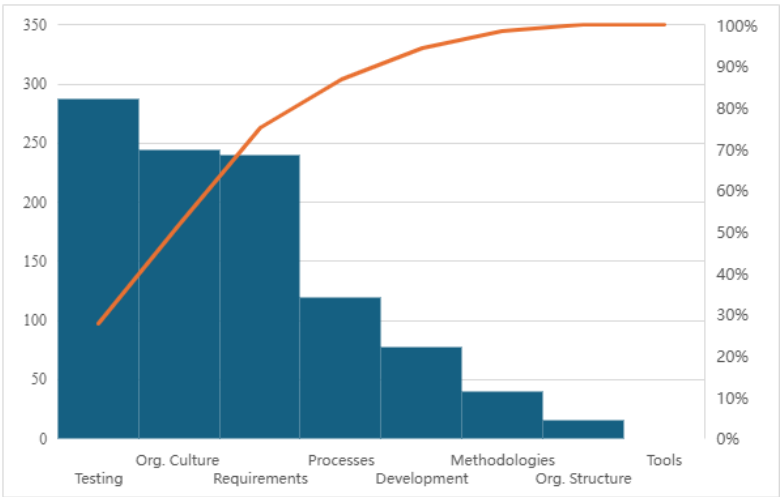

Figure 6 - Pareto Diagram with the analysis of the most impactful elements on SQA

According to Figure 7, the top three elements that impact the most software quality assurance are testing, organizational culture and requirements, since there is a noticeable difference between the slope of the accumulated frequency line and the values of the bars from the third element to the fourth. Being more specific, there is a split 37.5-75.2, which means that 37.5% of the elements represent 75.2% of the bottlenecks' total weight and that if these top three elements' respective factors are solved, 75.2% of the bottlenecks are solved as well, which

can be seen in more detail in Table 7. Therefore, associating with the factors inside each element, it is also possible to determine the most impactful factors.

For the final question, when asked about the most impacted areas soon by Artificial Intelligence, after calculating the average of the points given by every person to each category, the final ranking starts with a tie between coding and testing, with an average score of 4.2. The following most impacted areas are as shown in Table 8.

Table 8 - Ranking of the most impacted software development areas by Artificial Intelligence

	Average	Rank
Standardization	3,8	4
Product customization	3,86	3
Coding	4,2	1
Testing	4,2	1
Project management	3,12	6
Requirements	3,04	7
Follow up metrics	3,74	5

4.2 Results' Discussion

After the analysis of the results from the survey, it was possible to establish a connection between these results, the output of the interviews and the literature review previously performed. The script for the survey was designed based on the output from a previous related study and on the information given by the interviewees in several topics. Therefore, it was important to understand whether the reality extrapolated from the interviews actually applies to the general reality for companies in this sector.

When it comes to the factors preventing companies from achieving SQA, "poorly defined customers' requirements" was the option with more votes from the survey. Likewise, it was a problem widely discussed in the literature and from the four interviews, two of them mentioned that this was a bottleneck. Therefore, it is safe to confirm that this is indeed one of the most impactful factors identified.

On the other hand, "definition of quality standards" was only mentioned in one of the interviews, despite being the second most impactful factor determined by the survey. Additionally, in the literature review it was noted that ensuring quality standards would bring satisfactory results within the scope of software quality, but it was not a vastly explored topic the

other way around. This can show that there is not a lot of research about this topic and on how it can affect software quality when it is not ensured, despite happening very often inside the companies and being an important bottleneck, which can constitute an important study for the future.

The communication with the customer through the development process, being related to the requirements' issue (since the communication is meant for showing the clients the work done and collecting feedback on it) also showed up in the literature as a highly discussed problem. In agreement with the high importance and developed study about the topic, the interviews that covered the bottleneck of requirements also mentioned this matter. Even though this is a problem that can be diminished through ongoing alignments and calibration, it will always be a factor that can prevent companies from achieving SQA, unless there is habituation on the type of communication existent from both sides.

The most relevant remaining factors belonging to the elements that determine the most the success of software quality were: the need for improvement in code reviews (testing), the independence of team members (organizational culture), manual testing (testing) and ignoring the best quality practices (organizational culture). Despite the answers in the survey regarding code reviews, none of the interviews mentioned this problem, yet the literature included a lot of available information on the topic, which can only reveal that the companies of the people interviewed simply were not examples for this specific situation. The same happened exactly for manual testing. On the other hand, one of the interviews covered both the independence and the fact that professionals ignore the best quality practices, which, allied to the answers and the literature, demonstrate that these are often issues that affect companies of all sizes and needs to be worked on. These are both problems related to the culture of the company and the type of mentality associated with it.

Speaking of Quality, one of the other focuses of the study was if the culture of the companies is Quality related and if it fosters an environment in which professionals of the company are compelled to follow it. According to the answers of the questionnaire related to this and the number of votes in the factor "Lack of a culture of quality", it can be affirmed that there is still a significant percentage of companies that need to improve their organizational culture. A relevant number of people still do not identify some simple practices as important to foster this type of culture and others do not think the current implemented initiatives are contributing for it, so the higher levels of the company need to pay more attention to this topic. This is also a field that needs to be better researched, since the literature available did not give a lot of focus to it.

Time pressure has received a lot of attention from the literature and was also selected most of the time by the people answering the survey, being also mentioned in the interviews. Therefore, it is indeed one of the important factors, despite not being part of the most important elements. Usually, these types of decisions are not the developing teams' responsibility, so the higher hierarchy levels of the companies determine a lot the success of the projects by making decisions that are not the most appropriate for Quality. Yet, these are still important to ensure sales and the continuation of the business, so there needs to be a compromise to attain both. The same reality applies to priorities management, excluding the high focus from the literature, but with the additional fact that in Agile the people in contact with the customer may not be the ones developing the software. This could damage the final product based solely on the customers' will and possible ignorance on the features that should be a priority at certain times, to ensure quality software. Limited resources can also be included in this situation.

Another topic that showed coherence between the interviews and the literature review was data quality. Even so, it was not one of the most voted factors in the survey, not even belonging to the top three elements. This is a subject that has been investigated and is present in the literature, because it is a problem associated with small and less developed companies, which may be why it was not found in this study's questionnaire, which had a greater participation of bigger companies.

The least recognized problem in the survey was the reality of having teams working in distant locations, despite existing an interview discussing this issue and a couple of articles in the literature covering it as well, which at first indicated that this would be a more relevant problem that what it was showed. This situation suggests that, while it may not happen often, it could have been more common in the past, which is why it has already been studied and explored. A problem associated with this reality is the lack of coordination between developers and testers than can appear when these two entities are the ones in the globally distributed context. However, the number of votes in the survey for this bottleneck was three percentual points above the previous mentioned issue, which demonstrates that even working in the same space, there can be alignment problems, since the work of one group will have a direct influence on the other. Despite this reality, the lack of alignment is not covered by the literature as much as the in the globally distributed software context.

5

Conclusion

The objectives and milestones established to execute this dissertation were fully complete, which means that: 1) the context and nature of software development and software quality assurance was understood, through research on the different topics and by interviewing professionals employed in IT related companies, 2) a questionnaire that could comprehend how quality practices are spread and applied in these organizations was developed, 3) the results of the survey were obtained and analysed, and 4) the most impactful barriers and limitations to the project in IT companies were identified.

Even so, having in mind the research questions defined for the present study, it is possible to state that the outcome of the research could have been different if the number of answers collected for the questionnaire had been larger. Nonetheless, the data obtained allowed to answer, even with threats of invalidity to the research questions defined at the beginning of the investigation.

5.1 Results' Outcome

Starting by Question 1, presented in the Introduction of this document, it was possible to conclude that, even though this is an industry that incorporates a wide range of topics associated with Quality and with a lack of standard operating procedures due to the very different projects that can appear, most of the companies still describe Quality based on ISO Standards, so standardized regulation for projects and companies. This conclusion was not obtained through an absolute majority, but it was still a difference of 11%, which comparing to the rest of the sample was quite significative.

When it comes to Question 2, it was noticed that the companies' size is proportional to the human resources allocated to Quality, i.e. if the company had more employees, it was more likely to have a larger team or an entire department dedicated to this subject, while companies with less employees had a single person or a smaller team responsible for Quality assurance. Since there was a bigger number of answers from people that work in companies with more than 500 employees, the majority also had a large Quality team or a Quality department.

Regarding Question 3, it was confirmed that the most implemented metric is the percentage of compliance with customers' requirements, followed by customers' NPS and time. This allowed to conclude that customers' satisfaction metrics are more used by companies, despite the very similar percentage for each type of possible implemented metrics. There was also no difference on the type of methodologies implemented based on the number of employees.

Finally, and to fulfil the main purpose of this study, the most important elements preventing companies from achieving software quality were obtained, being these testing, organizational culture and requirements. Consequently, it can be affirmed that the most impactful factors are poorly defined customers' requirement, communication with the customer through the development process, independence of the team members which means lack of a standard procedure, poor definition of quality standards, manual testing, teams ignoring the best practices, planning the testing and validation phases, lack of a culture of quality and improvable code reviews. In any case, if there is the need to focus on the highly important three factors, despite the small differences of percentages that exist between them when their respective element is not considered, they are poorly defined customers' requirement, poor definition of quality standards and communication with the customer through the development process.

Additionally, and as a way to open the door for possible future studies regarding Quality in the IT industry, the data taken from the last question of the questionnaire can be used.

5.2 Barriers and Limitations

The initial barrier for this investigation resided in the fact that there are not a lot of open-source articles available in the indexation bases on some topics, for example, specific SQA bottlenecks addressed. This way, even though the present study becomes relevant through the addition of information to the existent research, it can also lack important theoretical context in the first half of the document. Consequently, the elaboration of the questionnaire could have missed important methodologies, bottlenecks or general factors that impact

SQA that people did not add on their own, choosing solely the option of "Other" in different questions, since they could have thought it was not relevant for this research.

On the other hand, the desired target for the number of answers was 100, since every company has an IT department, even if the main business focus is not software related, so with such a big IT universe, at least 100 perspectives could be a proper starting point for this research. Despite the efforts made to achieve this goal (described in Methodology), there was resistance from the people in this sector to respond to the questionnaire, which could invalidate some results, because if a larger number of people had answered towards an entirely different direction, the conclusions obtained from the data would have been different.

In the future, the same questionnaire can be applied again to understand the possible variation that may exist with a larger number of answers and clarify with more certainty the responses to the research questions. However, it is important to have in mind that the outcomes of any similar study will always be dependent on the time and context in which they are realized, since SQA practices are in constant evolution, as well as the IT industry.

BIBLIOGRAPHY

[1] H. Chen, M. Baptista Nunes, L. Zhou, and G. Chao Peng, "Expanding the concept of requirements traceability: The role of electronic records management in gathering evidence of crucial communications and negotiations," *Aslib Proc.*, vol. 63, no. 2/3, pp. 168–187, 2011, doi: 10.1108/00012531111135646.

[2] J. Faustino, D. Adriano, R. Amaro, R. Pereira, and M. M. Da Silva, "DEVOPS benefits: A systematic literature review," *Softw. Pract. Exp.*, vol. 52, no. 9, pp. 1905–1926, 2022, doi: 10.1002/spe.3096.

[3] N. Fatima, S. Nazir, and S. Chuprat, "Knowledge Sharing Framework for Modern Code Review to Diminish Software Engineering Waste," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 6, 2020, doi: 10.14569/IJACSA.2020.0110656.

[4] A. Jabborov, A. Kharlamova, Z. Kholmatova, A. Kruglov, V. Kruglov, and G. Succi, "Taxonomy of Quality Assessment for Intelligent Software Systems: A Systematic Literature Review," *IEEE Access*, vol. 11, pp. 130491–130507, 2023, doi: 10.1109/ACCESS.2023.3333920.

[5] D. Kato and H. Ishikawa, "Quality Control Methods Using Quality Characteristics in Development and Operations," *Digital*, vol. 4, no. 1, pp. 232–243, 2024, doi: 10.3390/digital4010012.

[6] S. Najihi, S. Elhadi, R. A. Abdelouahid, and A. Marzak, "Software Testing from an Agile and Traditional view," *Procedia Comput. Sci.*, vol. 203, pp. 775–782, 2022, doi: 10.1016/j.procs.2022.07.116.

[7] N. Soukaina, M. Soukaina, and M. Abdelaziz, "SQrum: An Improved Method of Scrum," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 9, 2022, doi: 10.14569/IJACSA.2022.0130928.

[8] Prof. S. D. Prof. Prashant Keswani, "Software Testing Methodologies: A Information Review," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 9, no. 2, pp. 01–05, 2021, doi: 10.17762/ijritcc.v9i2.5455.

[9] F. Almeida, J. Simões, and S. Lopes, "Exploring the Benefits of Combining DevOps and Agile," *Future Internet*, vol. 14, no. 2, p. 63, 2022, doi: 10.3390/fi14020063.

[10] H.-C. Estler, M. Nordio, C. A. Furia, B. Meyer, and J. Schneider, "Agile vs. structured distributed software development: A case study," *Empir. Softw. Eng.*, vol. 19, no. 5, pp. 1197–1224, 2014, doi: 10.1007/s10664-013-9271-y.

[11] A. Wiedemann, M. Wiesche, H. Gewald, and H. Krcmar, "Integrating development and operations teams: A control approach for DevOps," *Inf. Organ.*, vol. 33, no. 3, p. 100474, 2023, doi: 10.1016/j.infoandorg.2023.100474.

[12] M. Gannon, "An agile implementation of SCRUM," in *2013 IEEE Aerospace Conference*, Big Sky, MT: IEEE, 2013, pp. 1–7. doi: 10.1109/AERO.2013.6497388.

[13] A. Al MohamadSaleh and S. Alzahrani, "Development of a Maturity Model for Software Quality Assurance Practices," *Systems*, vol. 11, no. 9, p. 464, 2023, doi: 10.3390/systems11090464.
[14] A. Alami and O. Krancher, "How Scrum adds value to achieving software quality?," *Empir. Softw. Eng.*, vol. 27, no. 7, p. 165, 2022, doi: 10.1007/s10664-022-10208-4.
[15] T. Khalane and M. Tanner, "Software quality assurance in Scrum: The need for concrete guidance on SQA strategies in meeting user expectations," in *2013 International Conference on Adaptive Science and Technology*, Pretoria, South Africa: IEEE, 2013, pp. 1–6. doi: 10.1109/ICASTech.2013.6707499.
[16] M. Kuutila, M. Mäntylä, U. Farooq, and M. Claes, "Time pressure in software engineering: A systematic review," *Inf. Softw. Technol.*, vol. 121, p. 106257, 2020, doi: 10.1016/j.infsof.2020.106257.
[17] H. Ghanbari, T. Vartiainen, and M. Siponen, "Omission of Quality Software Development Practices: A Systematic Literature Review," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–27, 2019, doi: 10.1145/3177746.
[18] E. Mkoba and C. Marnewick, "Conceptual Framework for Auditing Agile Projects," *IEEE Access*, vol. 8, pp. 126460–126476, 2020, doi: 10.1109/ACCESS.2020.3007874.
[19] F. Shull, R. L. Feldmann, C. Seaman, M. Regardie, and S. Godfrey, "Fully employing software inspections data," *Innov. Syst. Softw. Eng.*, vol. 8, no. 4, pp. 243–254, 2012, doi: 10.1007/s11334-010-0132-1.
[20] F. Huang and L. Strigini, "HEDF: A Method for Early Forecasting Software Defects Based on Human Error Mechanisms," *IEEE Access*, vol. 11, pp. 3626–3652, 2023, doi: 10.1109/ACCESS.2023.3234490.
[21] M. A. Subih *et al.*, "Comparison of Agile Method and Scrum Method with Software Quality Affecting Factors," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 5, 2019, doi: 10.14569/IJACSA.2019.0100569.
[22] I. Kayes, M. Sarker, and J. Chakareski, "Product backlog rating: a case study on measuring test quality in scrum," *Innov. Syst. Softw. Eng.*, vol. 12, no. 4, pp. 303–317, 2016, doi: 10.1007/s11334-016-0271-0.
[23] A. A. Al-Rababah, T. AlTamimi, and N. Shalash, "A New Model for Software Engineering Systems Quality Improvement," *Res. J. Appl. Sci. Eng. Technol.*, vol. 7, no. 13, pp. 2724–2728, 2014, doi: 10.19026/rjaset.7.592.
[24] H. V. Gamido and M. V. Gamido, "Comparative Review of the Features of Automated Software Testing Tools," *Int. J. Electr. Comput. Eng. IJECE*, vol. 9, no. 5, p. 4473, 2019, doi: 10.11591/ijece.v9i5.pp4473-4478.
[25] A.-M. Vadan and L.-C. Miclea, "Software Testing Techniques for Improving the Quality of Smart-Home IoT Systems," *Electronics*, vol. 12, no. 6, p. 1337, 2023, doi: 10.3390/electronics12061337.
[26] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *J. Syst. Softw.*, vol. 123, pp. 176–189, 2017, doi: 10.1016/j.jss.2015.06.063.
[27] R. Florea and V. Stray, "The skills that employers look for in software testers," *Softw. Qual. J.*, vol. 27, no. 4, pp. 1449–1479, 2019, doi: 10.1007/s11219-019-09462-5.
[28] P. Orviz Fernández, M. David, D. C. Duma, E. Ronchieri, J. Gomes, and D. Salomoni, "Software Quality Assurance in INDIGO-DataCloud Project: a Converging Evolution of Software

Engineering Practices to Support European Research e-Infrastructures," *J. Grid Comput.*, vol. 18, no. 1, pp. 81–98, 2020, doi: 10.1007/s10723-020-09509-z.
[29] Q. Yas, A. Alazzawi, and B. Rahmatullah, "A Comprehensive Review of Software Development Life Cycle methodologies: Pros, Cons, and Future Directions," *Iraqi J. Comput. Sci. Math.*, pp. 173–190, 2023, doi: 10.52866/ijcsm.2023.04.04.014.
[30] A. Saravanos and M. X. Curinga, "Simulating the Software Development Lifecycle: The Waterfall Model," *Appl. Syst. Innov.*, vol. 6, no. 6, p. 108, 2023, doi: 10.3390/asi6060108.
[31] N. Yahya and S. Sarah Maidin, "Hybrid agile development phases: the practice in software projects as performed by software engineering team," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 29, no. 3, p. 1738, 2023, doi: 10.11591/ijeecs.v29.i3.pp1738-1749.
[32] P. Debois, "Agile Infrastructure and Operations: How Infra-gile are You?," in *Agile 2008 Conference*, Toronto, ON, Canada: IEEE, 2008, pp. 202–207. doi: 10.1109/Agile.2008.42.
[33] N. M. Salleh and P. Ne, "Comparative Study between Lean Six Sigma and Lean-Agile for Quality Software Requirement," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 12, 2019, doi: 10.14569/IJACSA.2019.0101230.
[34] N. Goker and M. Dursun, "Evaluation of Project Management Methodologies Success Factors Using Fuzzy Cognitive Map Method: Waterfall, Agile, And Lean Six Sigma Cases," *Int. J. Intell. Syst. Appl. Eng.*, vol. 10, no. 1, pp. 35–43, 2022, doi: 10.18201/ijisae.2022.265.
[35] M. Aamir and M. N. A. Khan, "Incorporating quality control activities in scrum in relation to the concept of test backlog," *Sādhanā*, vol. 42, no. 7, pp. 1051–1061, 2017, doi: 10.1007/s12046-017-0688-7.
[36] T. Natarajan and S. Pichai, "Transition from Waterfall to Agile Methodology - An Action Research Study," *IEEE Access*, pp. 1–1, 2024, doi: 10.1109/ACCESS.2024.3384097.
[37] A. V. Jha *et al.*, "From theory to practice: Understanding DevOps culture and mindset," *Cogent Eng.*, vol. 10, no. 1, p. 2251758, 2023, doi: 10.1080/23311916.2023.2251758.
[38] N. Azad and S. Hyrynsalmi, "DevOps critical success factors — A systematic literature review," *Inf. Softw. Technol.*, vol. 157, p. 107150, 2023, doi: 10.1016/j.infsof.2023.107150.
[39] M. A. Akbar, A. A. Khan, N. Islam, and S. Mahmood, "DevOps project management success factors: A decision-making framework," *Softw. Pract. Exp.*, vol. 54, no. 2, pp. 257–280, 2024, doi: 10.1002/spe.3269.
[40] M. A. Pastrana Pardo, H. A. Ordoñez Erazo, and C. A. Cobos Lozada, "Documenting and implementing DevOps good practices with test automation and continuous deployment tools through software refinement," *Period. Eng. Nat. Sci. PEN*, vol. 9, no. 4, p. 854, 2021, doi: 10.21533/pen.v9i4.2239.
[41] R. Grande, A. Vizcaíno, and F. O. García, "Is it worth adopting DevOps practices in Global Software Engineering? Possible challenges and benefits," *Comput. Stand. Interfaces*, vol. 87, p. 103767, 2024, doi: 10.1016/j.csi.2023.103767.
[42] A. Mishra and Z. Otaiwi, "DevOps and software quality: A systematic mapping," *Comput. Sci. Rev.*, vol. 38, p. 100308, 2020, doi: 10.1016/j.cosrev.2020.100308.
[43] M. H. Tanzil, M. Sarker, G. Uddin, and A. Iqbal, "A mixed method study of DevOps challenges," *Inf. Softw. Technol.*, vol. 161, p. 107244, 2023, doi: 10.1016/j.infsof.2023.107244.
[44] R. A. Khurma, H. Alsawalqah, I. Aljarah, M. A. Elaziz, and R. Damaševičius, "An Enhanced Evolutionary Software Defect Prediction Method Using Island Moth Flame Optimization," *Mathematics*, vol. 9, no. 15, p. 1722, 2021, doi: 10.3390/math9151722.

[45] W. Alsaqaf, M. Daneva, and R. Wieringa, "Quality requirements challenges in the context of large-scale distributed agile: An empirical study," *Inf. Softw. Technol.*, vol. 110, pp. 39–55, 2019, doi: 10.1016/j.infsof.2019.01.009.

[46] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "An exploratory study on confusion in code reviews," *Empir. Softw. Eng.*, vol. 26, no. 1, p. 12, 2021, doi: 10.1007/s10664-020-09909-5.

[47] G. Rodríguez-Pérez, G. Robles, A. Serebrenik, A. Zaidman, D. M. Germán, and J. M. Gonzalez-Barahona, "How bugs are born: a model to identify how bugs are introduced in software components," *Empir. Softw. Eng.*, vol. 25, no. 2, pp. 1294–1340, 2020, doi: 10.1007/s10664-019-09781-y.

[48] M. Gan, Z. Yucel, and A. Monden, "Improvement and Evaluation of Data Consistency Metric CIL for Software Engineering Data Sets," *IEEE Access*, vol. 10, pp. 70053–70067, 2022, doi: 10.1109/ACCESS.2022.3188246.

[49] M. Brunetto, G. Denaro, L. Mariani, and M. Pezzè, "On introducing automatic test case generation in practice: A success story and lessons learned," *J. Syst. Softw.*, vol. 176, p. 110933, 2021, doi: 10.1016/j.jss.2021.110933.

[50] A. Panichella, S. Panichella, G. Fraser, A. A. Sawant, and V. J. Hellendoorn, "Test smells 20 years later: detectability, validity, and reliability," *Empir. Softw. Eng.*, vol. 27, no. 7, p. 170, 2022, doi: 10.1007/s10664-022-10207-5.

A

APPENDIX

Questionnaire

1. What is your age?

- 18 - 25 years old
- 26 - 35 years old
- 36 - 45 years old
- 46 - 55 years old
- 56 - 65 years old
- +65 years old

2. What are your academic qualifications?

- High school
- Bachelor
- Master
- PhD
- MBA
- Other:

3. What is your role inside the company?

- Developer
- Tester
- Quality Assurance Analyst (or similar)
- Quality Assurance Engineer

- Quality Manager
- Product Manager
- Product Owner
- Project Manager
- Project Owner
- Other:

4. For how long have you been an employee at the company?

- 0 - 5 years
- 5 - 10 years
- 10 - 20 years
- 20 - 30 years
- +30 years

5. How many years of experience do you have in IT functions?

- 0 - 5 years
- 5 - 10 years
- 10 - 20 years
- 20 - 30 years
- +30 years

6. Is the core of your company IT or are you part of a department supporting IT operations?

- IT Department
- IT Company

7. What is the company's core activity?

- Software Development
- Software platform sales/management
- Other:

8. How would you best describe your company?

- Start Up
- Scale Up
- Established player

- National market leader
- Regional/Global market leader

9. What is the number of employees inside the company (at your site)?

- 1-10
- 10 - 50
- 50 -250
- 250 - 500
- 500+
- 2500+

10. Where is the company situated?

- Asia
- Africa
- Europe
- Middle East
- North America
- South America

11. What would best describe the understanding of Quality within your company?

- ISO Standards
- Customer's requirements fulfilment
- Stakeholders' satisfaction
- Specific project metrics
- Specific metrics established by the Quality department/team
- Density of errors/defects
- Number of complaints from customers
- Other:

12. Who is responsible for Quality inside the company?

- One single person
- Small team (2 – 4)
- Large team (5 – 8)
- Quality task force (8+)

- Quality Department

13. (if there is a team or group of people) What are these people part of?
    - One single team, each person following multiple projects.
    - One single team, each person following different project stages.
    - Each person is allocated to a specific project.
    - A smaller group is allocated to specific projects.
    - Other (please describe):

14. How does your company manage projects and guarantee quality?
    - Traditional: Waterfall
    - Traditional: other
    - Agile: Scrum
    - Agile: XP
    - Agile: other
    - DevOps
    - Other:

15. Which type of metrics are used to measure Quality?
    - Project related metrics
    - Customer satisfaction related metrics
    - Product related metrics

16. (depending on the previous answer) Which metrics are used to measure Quality?
    - Time (project)
    - Cost (project)
    - Delay (project)
    - NPS (customer satisfaction)
    - %Compliance with customer requirements (customer satisfaction)
    - #CodeSmells (product)
    - #Vulnerabilities (product)
    - Defects Density (product)
    - Other:

17. Would you say there is a culture of Quality in your company?

- Yes
- No

18. (if the previous answer was yes) Does this culture of Quality vary according to the department of the company?

- Yes
- No

19. (if the answer to the first question of this section was yes) What better characterizes this culture of Quality? (pick 3)

- Outspoken focus on Quality across corporate communication
- Development of dedicated metrics, tools or models to manage Quality
- Quality Awards or recognition from customers & stakeholders
- Training
- Other (please describe):

20. Is it common for employees to discuss the best practices together for their tasks?

- Yes
- No

21. What type of initiatives related to Quality are implemented by the company's leadership?

- Ongoing workshops related to the topic
- Certified quality related courses
- Sharing of best practices from different teams
- Feedback culture through specific platforms and practices
- Individual development plan for the employees
- Other:

22. What do you consider to be the 5 main factors that may be preventing your company from having better Quality assurance/management?

- Lack of a culture of quality
- Limited resources
- Poorly defined customers' requirements

- Communication with the customer through the development process
- Lack of coordination between developers and testers
- Teams working in very distant locations, mainly developers and testers
- Priorities management
- Quality assurance procedures not adapted to software development projects
- Code reviews could improve
- Data quality from previous projects damages the planning of new ones
- Definition of quality standards
- Planning the testing and validation phases
- Independence of the team members, meaning there is no standard, and everyone uses their own experience
- Manual testing
- Time pressure
- Ignoring the best practices
- Other (please describe):

23. Choose the 3 elements that are more specific regarding quality in IT.

- Requirements
- Development
- Testing
- Processes
- Methodologies
- Tools
- Organizational structure
- Organizational culture

24. From 1 to 5, how do you think each of these quality components will be impacted by Artificial Intelligence?

- Standardization
- Product customization
- Coding
- Testing
- Project management
- Requirements

- Follow up metrics