User Acceptance Testing Process for a Digital Public Service

António Jorge Novais Vidal Fernandes

Thesis to obtain the Master of Science Degree in

Industrial Engineering and Management

Supervisors:
Prof. Miguel Leitão Bignolas Mira da Silva
Prof. Tânia Rodrigues Pereira Ramos

Examination Committee

Chairperson: Prof. Susana Isabel Carvalho Relvas
Supervisor: Prof. Tânia Rodrigues Pereira Ramos
Member of the Committee: Prof. Rita Andreia da Conceição Marques

June 2023

Acknowledgements

I would like to express my gratitude to my supervisors Professor Miguel Leitão Bignolas Mira da Silva and Professor Tânia Rodrigues Pereira Ramos for the constant support, guidance, and availability throughout this process. Thank you for sharing your expertise and providing me with constructive feedback.

I would also like to extend my appreciation to Filipe Moreira, Teresa Carvalho, and all the personnel of Agência Nacional de Inovação that did not hesitate to be helpful and available throughout this research. Your support greatly contributed to the result of this Master's Thesis.

Finally, my wholehearted thanks to my parents, to my sister Joana, and to my girlfriend Rita for the endless support, patience, and encouragement during challenging times throughout this academic journey. Thank you for everything.

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Abstract

User Acceptance Testing is a crucial phase in software development that involves evaluating software from the end-users' perspective to ensure that it satisfies their needs and expectations. Therefore, it is a process that needs to be performed accurately and consistently for correct software implementation.

A comprehensive user acceptance testing process for digital public services enhancing efficiency and effectiveness is developed in collaboration with a Portuguese digital public service agency responsible for promoting innovation and technology transfer - Agência Nacional de Inovação. Challenges were identified in implementing digital public services software due to inadequate user acceptance testing processes. Hence, this research aims to develop a tailored process that would suit their characteristics and improve the success rate of software implementation, with Agile principles and enhancing the Scrum Framework.

Supported by a literature review on the topic, the tailored process was tested through real-world cases, involving two software applications developed by Agência Nacional de Inovação. User feedback through surveys and interviews was effectively incorporated into the testing process. The results showed significant improvements in efficiency and effectiveness compared to previous testing processes used by the Agency, with issues found in more than 20% of the tested user stories that have already undergone user acceptance testing.

This research contributes to the current body of knowledge on user acceptability testing by offering a novel perspective, as well as helpful insights and advice for organizations and businesses who supply software with similar purpose.

Keywords: user acceptance testing, software development, digital public service, process, agile, scrum

Resumo

Testes de aceitação do utilizador são cruciais no desenvolvimento de software que envolve a avaliação do mesmo na perspetiva dos utilizadores finais para garantir que satisfaz as suas necessidades e expectativa. Como tal, esse processo tem de ser preciso e consistente para uma correta implementação do software.

Foi desenvolvido um processo de testes de aceitação do utilizador para serviços públicos digitais em colaboração com uma agência pública portuguesa de promoção da inovação e transferência de tecnologia, focado em eficiência e eficácia, e identificados obstáculos na implementação de software devido a processos inadequados de testes de aceitação do utilizador. Assim, esta investigação visa desenvolver um processo à medida que se adeque às suas características e melhore a taxa de sucesso da implementação de software, contendo princípios Agile e reforçando a Framework Scrum.

Partindo de uma revisão da literatura sobre o tema, o processo desenvolvido foi testado em casos práticos, envolvendo duas aplicações de software desenvolvidas pela Agência Nacional de Inovação. O feedback dos utilizadores foi efetivamente incorporado no processo de testes através de inquéritos e entrevistas. Os resultados mostraram melhorias significativas na eficiência e eficácia em comparação com os processos de teste anteriormente utilizados pela agência, com problemas encontrados em mais de 20% das histórias de utilizador testadas que já foram submetidas a testes de aceitação do utilizador.

Esta investigação contribui para o atual corpo de conhecimentos sobre testes de aceitação de utilizador, oferecendo uma perspetiva diferente, bem como ideias úteis para organizações e empresas que fornecem software com propósito semelhante.

Palavras-chave: testes de aceitação do utilizador, desenvolvimento de software, serviço público digital, processo, agile, scrum

Table of Contents

List of Tables

List of Figures

Acronyms

ANI – *Agência Nacional de Inovação*

BIP – Business Information Platform

DIPI - *Direção de Incentivos e Programas Internacionais*

IS – Information Systems

IT – Information Technology

POAT - *Programa Operacional de Assistência Técnica*

SGI – *Sistema de Gestão de Incentivos*

SGP – *Sistema de Gestão de Peritos*

SIFIDE – *Sistema de Incentivos Fiscais à Investigação e Desenvolvimento Empresarial*

UAT – User Acceptance Testing

1. Introduction

1.1 Background and Motivation

Project management methodologies are essential components of software development, providing key guidelines that enable project teams to effectively navigate various situations and challenges. In recent years, a variety of methodologies were proposed by researchers that increased the efficiency of software development in contrast with Traditional Project Management (Royce, 1970), namely Agile approach and the Scrum Framework (Beck et al., 2001; Schwaber & Sutherland, 2020), which empowered teams to work iteratively, allowing them to quickly respond to change.

DevOps practices (de França et al., 2016) and Product Management (Cagan, 2017) have also contributed to an increase in the diversity of perspectives, enabling teams to adapt to a broader range of scenarios based on the product or the team composition they provide.

However, with that rapid and iterative development, it comes the need to effectively test the application along with its development, which it is not easy to achieve in the Agile or Scrum context (Padmini et al., 2016). The current state of investigation on the topic is very disperse, meaning that there is not a clear solution for that obstacle, that being as important as the software development itself, as bugs or defects can be found in a later stage, possibly compromising user experience and ultimately the company's image.

Consequently, this must be addressed by developing a process that is both more effective and efficient and by offering tools that are tailored to the employees that will execute it.

This dissertation was developed in the context of an internship as Project Management assistant at the Information Systems (IS) Department of *Agência Nacional de Inovação* (ANI).

ANI – or National Innovation Agency - is an entity that focuses on enhancing the innovation process in Portugal. It promotes the collaboration between the scientific and technological fields along with the corporate world.

The Agency encourages the participation of Portuguese entities in international programs, finances research and development programs through financial and tax incentives, and aids in the dissemination of innovation on both a national and international scale.

ANI uses a variety of tools to handle and evaluate applications from a wide range of companies to receive both tax and financial incentives from investments in innovation and participation in international programs. It also takes measures to ensures that innovation is continuous in the country, by linking corporate to specific entities, such as universities, promoting knowledge sharing that contributes for the

enrichment of both parties. To uphold and optimize ANI's mission, the IS Department assumes a relevant role in the in the digital transformation and transition it has set out to achieve.

For that reason, the National Innovation Agency has a dedicated IS department that handles everything that concerns the digital platforms that support those practices, including software development and maintenance, both outsourced and in-house.

The IS Department has a varied portfolio of software development projects. They manage these projects through an adaptation of Scrum methodology and, as any firm in the field has its nuances and practices. The software developed must be tested not only by the development team but also by the business users in ANI, through User Acceptance Testing (UAT), which must be adapted to their needs and the software development methodology itself. For that reason, performing UAT is, therefore, a challenging task for the organization – although it occurs, there is not a defined set of procedures that the company personnel must follow in order to make sure that UAT is conducted correctly.

For the aforementioned reasons, the motivation for the present dissertation is the development of a UAT Process that suits ANI's needs. This research is important as it will allow the firm to deploy better, more capable applications for its users, avoiding potential problems mid software development cycle, and ultimately contributing for the better functioning of the company as a whole.

1.2 Research Objectives

The main objective of this dissertation is to develop a UAT Process for ANI. There are, however, secondary goals that were defined and will help to contribute to the main research:

- Characterize ANI, namely how its departments are interconnected, with focus on the IS department.
- Understand the type of software development projects in which the company is involved.
- Comprehend the Project Management Methodology used at ANI, as well as the used technology for that end.
- Find out about opportunities for improvement in the IS department that could contribute to an efficiency and effectiveness improvement.
- Learn about the most used software development methodologies nowadays and how each one of them contributes for an improvement of affairs for the projects in which they are employed.
- Learn about previous research involving UAT, finding out about different perspectives given different contexts and how its lessons could be applied to the current problem.
- Provide a UAT process that could solve the concerns and objectives of this research in ANI.
- Test the process in current projects of the firm to find out about the feasibility and effectiveness applied to a wide range of projects.

- Educate and collect feedback from potential users of the process, taking into account the different views provided so that an improvement of the process could be made and how that could be applied to the company.
- Present evidence of the feasibility of the process by presenting approval records from users, as well as data that verifies the effectiveness of the project in the projects of ANI.

1.3 Research Approach

The approach for the research found in the present dissertation, represented in Figure 1, will comprise 5-steps to achieve the outlined objectives in Section 1.2:

1. Company context and Opportunities: Based on the experience gathered in the 8-month internship in ANI, the company context will be studied, involving its structure respresented by using the Archimate 3.2 modelling language, methodologies, current status, and opportunities for improvement, that can contribute to identify the main challenges and action areas for this study.

2. Software Development Methodologies Background: Study the most used Software development methodologies and how they compare between each other, outlining their advantages and disadvantages. That information can set new ideas and provide understanding for the following research.

3. Literature Review on UAT: Study used approaches for UAT in a range of contexts regarding the Agile methodology and understand how it contributed for an improvement of software testing in such circumstances. The research was done using the Google Scholar search engine with a set of predefined strings so that the focus of the research was not compromised, accessing a variety of databases. Further research was permitted by the usage of forward and backward snowballing techniques.

4. UAT Process Development: Define the process tailored for the needs of ANI using an iterative approach and enhanced by the usage of a flowchart built using the online software Miro (*The Visual Collaboration Platform for Every Team | Miro*, n.d.).

5. Process Validation, Experiments and Results: Organization of formal meetings and interviews with company personnel to validate the process, application of the proposed process in two ongoing projects, and qualitative results analysis regarding past UAT performances in the company.

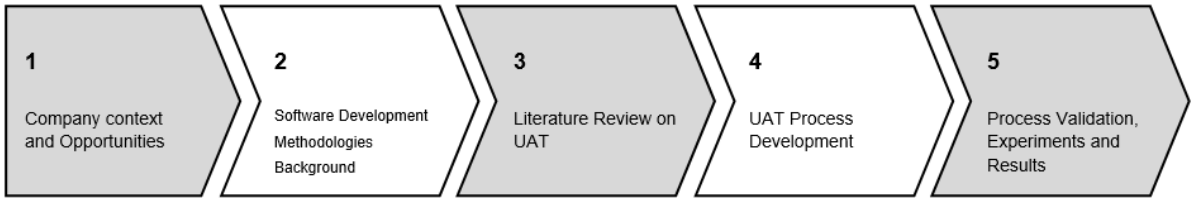

Figure 1 - Research Approach

1.4 Document Structure

Considering the given research approach, the document structure will consist of the nine following chapters:

1. Introduction: This chapter displays the background and motivation of the research, the objectives, the research approach, and the document structure.

2. Research Context: Contextualizes ANI, namely its organizational structure and the IS Department, the research perspective and highlights challenges that represent opportunities for improvement.

3. Research Background: Identifies significant software development methodologies and compares them across three distinct areas.

4. UAT Literature Review: Describes previous UAT research, illustrating how UAT was conducted in a variety of cases in the past, highlighting the key ideas.

5. Research Methodology: Defines the methodology utilized to perform the UAT iterative process given in the next chapters.

6. First Proposal: Describes the first literature-based proposal for the UAT process for ANI, as well as the experience acquired by conducting UAT on a software application for a project and the insights gained from meetings with the Product Owner.

7. Second Proposal: Defines the second experience-based proposal for the UAT process for ANI, as well as the experience obtained from executing UAT on a separate ANI portfolio project and the insights gathered through interviews with the product owners of the ANI portfolio projects.

8. Final Proposal and Discussion of Results: Describes the final UAT process proposal for ANI, and details the approval process based on interviews and UAT experience data analysis.

9. Conclusions: Summarizes the most significant conclusions of the research, and indicates its limitations and future research prospects.

2. Research Context

2.1 Organizational Structure of Agência Nacional de Inovação

In order to have a meaningful impact on a digital public service company, it is crucial to understand its structure right down to its core. The structure of ANI is traditional and straightforward. The department that is responsible for everything that occurs at the company is the Board of directors, which consists of the President, two directors, and board of directors' advisors, who understand the business and are able to present the board with a unique perspective on prospective upgrades. The board of directors is responsible for and supervises every other department in the organization, serving as sponsors and providing the means and vision for everything that occurs in those departments.

Using Archimate Full Framework following OneGroup 3.2 Specification (Introduction: ArchiMate® 3.2 Specification, n.d.) and Archi Software (*Archi – Open Source ArchiMate Modelling*, n.d.) for its modelling, ANI's Organizational Viewpoint is shown in Figure 2:

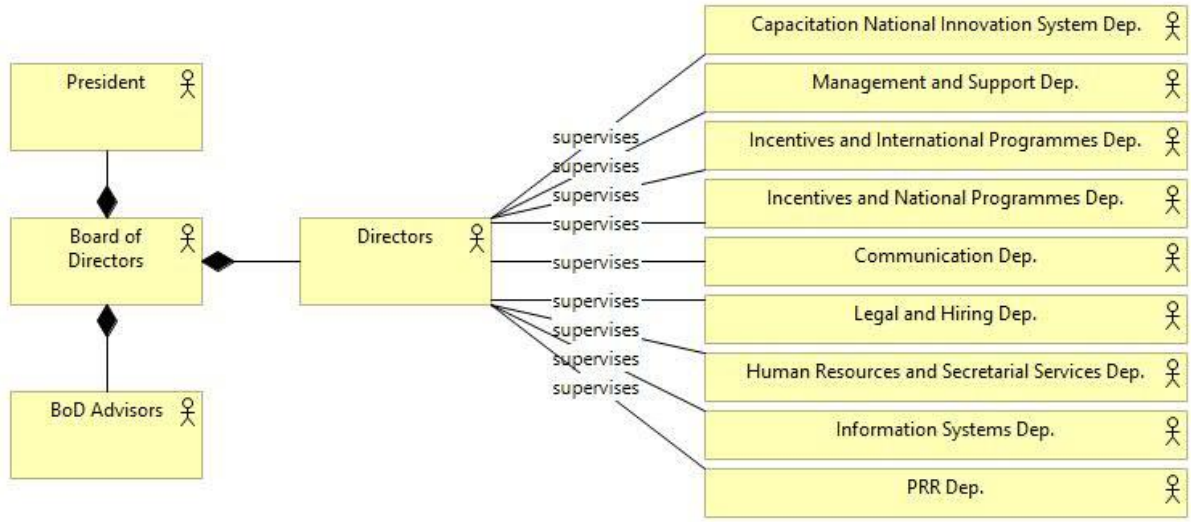

Figure 2 - ANI Organizational Viewpoint

ANI has nine departments – the Capacitation Nacional Innovation System department, Management and support department, Incentives and International Programmes department, Incentives and National Programmes department, Communication department, Legal and Hiring department, Human Resources and Secretarial Services department, IS department and Recovery and Resilience Plan department – crucial to the company's activity. They are complementary since every project that ANI is involved with

needs more than one department to fulfil its purpose. The IS Department will be the focus of this dissertation, as the research objectives pertain to this department.

2.2 Information Systems Department

2.2.1 Organizational Structure

The IS Department is supervised by a Head of IS department who oversees all operations and manages all efforts to achieve the department's objectives. A Project Manager is appointed to oversee the whole portfolio, reducing obstacles and entropy while reporting to the IS department head. Every project has a Product Owner who is liable for all project-related decisions, is in charge of the requirements, communicates frequently with the development team, and has the last say in the acceptance process.

The Agency's portfolio consists of a wide variety of initiatives designed to solve various problems. They are positioned differently within the organization and organized into three distinct programs based on their intended outcomes: Business Applications, Business Support and *Programa Operacional Assistência Técnica* (POAT), or Technical Assistance Operational Programme.

Business Applications consists of six projects that which are the heart of ANI's operation and are vital to the company's survival and realization of its goal. Its objectives include granting tax or financial incentives that allow the company to flourish, as well as the tools to input data that enable an evaluation of a company's research and development status. In addition, there are projects for platforms that facilitate the recruitment of experts to evaluate those applications, in case that ANI personnel lack sufficient expertise on certain topics, and a centralized website that includes the full range of contacts that a particular individual had with the agency, so that all information is readily available and none is lost in the process.

Business Support Applications are meant to help the employee's day-to-day tasks and are not intended for the use of external personnel. Included are portals for tracking and analysing personal costs, business travel, and budgeting, as well as a portal that an ANI employee may use to access any other platform related with their account, if access is authorized.

Lastly, Technical Assistance Operational Programme is a tool designed to manage archive-related questions. With the goal of converting every physical document to cloud-based, accessible documentation, a process is underway to ensure that the data included in this documentation is readily available.

Using Archimate Full Framework, the IS Department is represented in Figure 3:

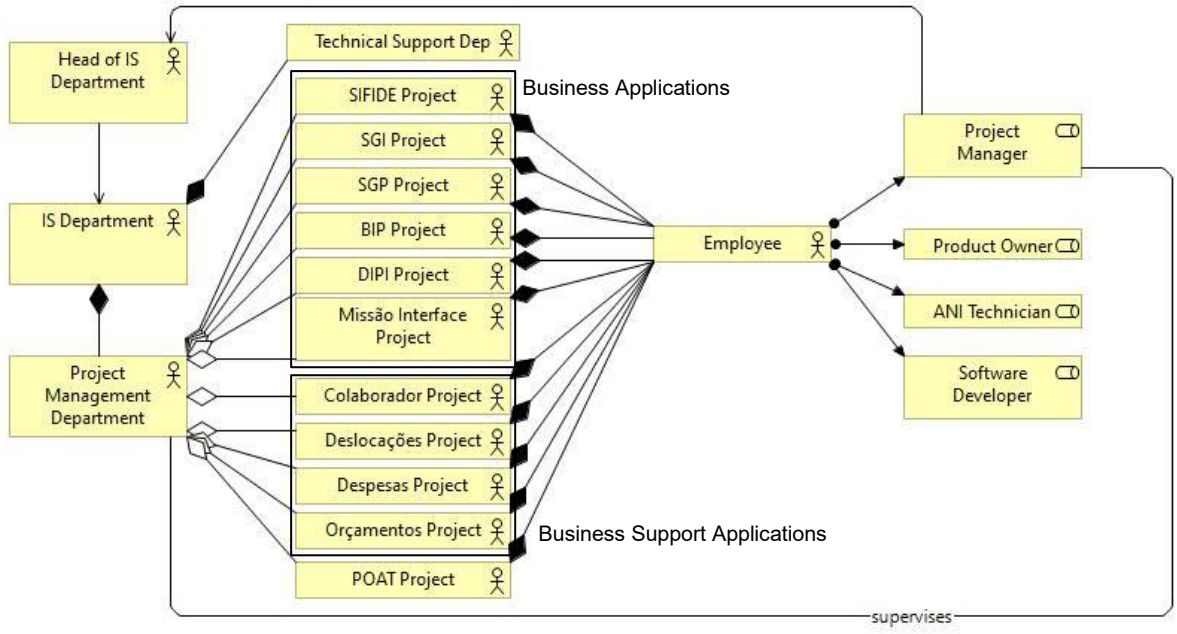

Figure 3 - IS Department Organizational Viewpoint

Every project – and application – referred in Figure 3 needs the collaboration of the ANI personnel for its development, and as future business users. Those employees – ANI technician, Product Owner and Project Manager and Software Developer all have their set of responsibilities and need to be coordinated so that the project flows efficiently.

Because of the quantity and diversity of the projects assigned, it is imperative that the information technology department, and software programs, operate as smoothly as is humanly feasible. This is the reason why the present research is being conducted; ANI's business operations depend on all of them.

2.2.2 Used Technology

ANI is continuously thriving to use the best possible technology so that their platforms present the least number of issues and can be easily managed or updated. For that reason, the development of those applications is restricted to four technologies:

- **Outsystems:** A low-code environment that uses drag-and-drop to ease development rather than countless lines of code, and is aimed to deliver applications into production considerably faster than usual (*What Is Low-Code | Low-Code Guide*, 2023).
- **ERP Primavera Public Sector v9:** Enterprise Resource Planning Software that provides a centralized management system tailored for public institutions (*ERP Public Sector SNC-AP - Software de Contabilidade Pública*, 2023).

- **.Net:** Also known as dotnet, is an opensource framework created by Microsoft to build and develop applications. (*..NET | Build. Test. Deploy.*, 2023)
- **PowerBI:** Interface data visualization software product (*Visualização de Dados | Microsoft Power BI*, 2023)

ANI now outsources most development tasks, apart from the DIPI and Interface Mission projects. The remainder of the apps are being developed by an external supplier, in coordination with the Agency, utilizing Outsystems for either development from scratch or migration of existing platforms.

The Interface Mission software is being created by an internal employee utilizing the .Net framework given the experience with the platform, since maintenance will be performed internally.

PowerBI is employed as the principal technology for the DIPI project, as its purpose involves data processing and presentation of archived data, although the website is developed using Outsystems.

2.2.3 Project Management Methodologies

The IS Department employs a hybrid technique for project management. It combines Agile and Scrum concepts and a Kanban board to monitor the development processes.

The roles present in the employed methodology are based on the roles present in the Scrum Methodology; the Project Manager incorporates the roles of a Scrum Master and has additional responsibilities related to traditional project management approaches, including the preparation of reports and documentation, planning and scheduling activities, and risk management. There are pre-scheduled meetings with the supplier, similar to the notion of a Sprint, to assess the work completed within a specific time period. The Product Owner is also responsible for managing the Product Backlog for requirements management, using the user story format recommended by the Scrum Guide (Schwaber & Sutherland, 2020).

Utilizing a customized Kanban Board, represented in Figure 4, with columns labelled Not Ready, Backlog, In Progress, Blocked, In Acceptance, In Testing, and Completed allows the management of operations and development. Thus, the supplier and ANI personnel are in continual sync with respect to the development status of the project, by dragging and dropping different stickers based on the status of a particular user story. The Board is driven by Microsoft Planner (*Microsoft Planner*, 2023), which provides a straightforward interface that can be readily controlled by all concerned parties. The program supports notes, checklists, and comments, but lacks complex prioritizing techniques and data exporting tools to better depict the project's status, making it difficult to follow. Consequently, compromises were made as simplicity was favoured.

The Agency favours the Agile Manifesto for Software Development Principles (Beck et al., 2001), by endorsing customer collaboration by maintaining constant contact with the supplier, as well as by

responding to change rather than strictly adhering to a plan, by iteratively revaluating requirements and engaging in development activities.

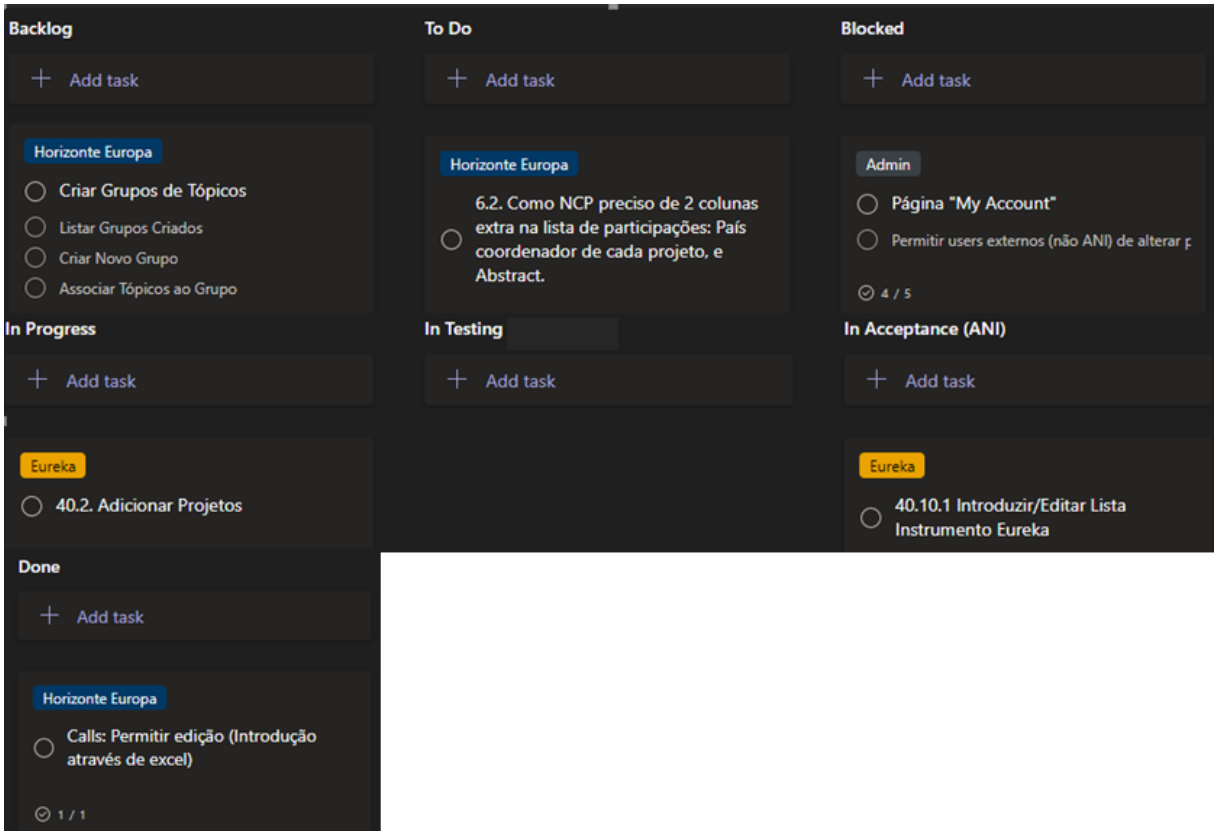

Figure 4 - Planner Kanban Board

The general Software Development Business Process Viewpoint in ANI is as shown in Figure 5:

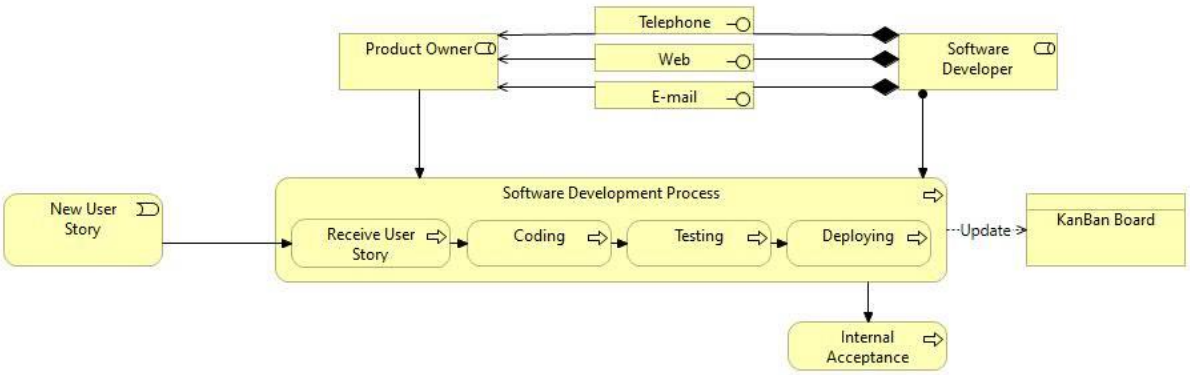

Figure 5 - Business Process Viewpoint

When a new user story is created, the software development process begins with the development team or supplier receiving the user story by viewing the Kanban Board. The development team then executes

the processes of coding, testing, and deployment. Subsequently, business users of ANI must perform acceptance testing of the software deployed, so it fulfils the set requirements. All of this is accomplished by the regular updating of the Kanban board by the development team and Product Owner of the project, while the development team and Product Owner maintain constant contact via telephone, online meetings, and email.

2.3 Research Perspective

This dissertation is part of an IT Internship in IS Department at ANI. The initial phase of the internship was to comprehend not only the company, but also the reasons behind everything that is done at ANI, in order to generate more meaningful and focused work.

The internship function was Project Management Assistant. The responsibilities included assisting the Project Manager with everything related to the ongoing projects, such as scheduling meetings with team members and suppliers, eliminating barriers identified by the product owner, risk assessment and management through identification with the Product Owner, preparing timed reports for the Agency's board of directors, and educating through a more effective application of agile principles, among others.

There was daily participation in team meetings, either with the team in charge of a specific project, with the team and supplier responsible for software development, or with the product owners of the various products, during which they shared valuable knowledge. These meetings facilitated a constant flow of feedback among all participants, maintained a healthy environment for continuous improvement, and guaranteed that each process step was documented.

On occasion, work was also assigned based on the predominant area of expertise., mostly in the IS department, but also in other areas. Small-scale projects included the use of Microsoft Excel, Microsoft Power Point, and Microsoft Power Automate (*Power Automate | Microsoft Power Platform*, n.d.; *Software de Apresentação de Diapositivos do Microsoft PowerPoint | Microsoft 365*, n.d.; *Software de Folha de Cálculo do Microsoft Excel | Microsoft 365*, n.d.). These were mostly related with data extraction from files, but also with evaluating the performance of the software development supplier by determining the amount of wrongly invoiced hours and enabling a more cohesive and accurate accounting of departmental expenditures.

The direct engagement in the firm's operations helped in getting a more realistic picture of the organization, which exposed all the hurdles that exist in such a department and allowed to connect with personnel with diverse backgrounds, personalities, and motivations. The position demanded a variety of responsibilities, which helped to raise awareness for the department's current capability but also its challenges, not only in terms of Project Management, but also to characterize the overall environment at ANI. Section 2.4 will help to identify such difficulties, but also potential for improvement at ANI.

2.4 Opportunities for Improvement

In this section, opportunities for improvement are described. To grasp the current condition of software development at ANI, a survey was developed for Product Owners to identify the opportunities presented by the personnel in charge of the projects, and an analysis was conducted to determine the number of defects discovered in the relevant projects at the time.

2.4.1 Survey to Product Owners

The daily expertise obtained at ANI was essential to grasp the current situation. It is also crucial to include the perspectives of those who interact with the projects on a regular basis - the Product Owners of each project.

A survey was prepared and disseminated to all Product Owners in order to identify and comprehend the issues and obstacles being encountered.

It was determined that these questions would accurately reflect the status of the whole collection of projects managed by the IS Department. The survey consisted of a total of thirty-four questions, eighteen of which answered through a five-point Likert scale, a simple, yet effective way to determine participants' degree of agreement or disagreement, satisfaction or dissatisfaction with a particular statement (Bertram, 2006), having as possible answers strongly disagree, somewhat disagree, neutral, somewhat agree and strongly agree, ordered in ascending order of concordance. Additionally, sixteen open-ended questions were posed, some of which added to the prior question if a negative response was given. The majority of the questions were motivated by this research, some were requested by the IS Department.

The survey received seven responses, which represented the total number of Product Owners assigned to projects at the time it was sent. Appendix A – Survey Questions to the Product Owners contains all the questions and answers for the Likert-scale questions. Although there are eleven projects depicted in Figure 3, one of them is lacking a product owner, and some are managed by the same Product Owners.

The survey consisted of three groups of questions, as follows:

- As a Product Owner
- About the Supplier’s Team
- About the Project Management Team

The approach developed by Dane Bertram (Bertram, 2006) for analysing Likert Scale surveys could be utilized in this research. However the number of responses is not enough for a descriptive statistical analysis, even though all the possible participants have completed the survey. As a result, a qualitative

analysis is conducted, showing the essential insights of these replies, highlighting the central concepts of each question group.

As a Product Owner:

This set of questions were designed to define the Product Owners and their performance, as well as the status of the project and any obstacles that may be jeopardizing its success.

This group's responses indicated that they had a positive perception of their performance and motivation as Product Owners, as well as a general sense of support in the execution of the assigned project. However, the majority of Product Owners believe that the project has not been proceeding as planned, citing the small amount of time allocated to the project. All the Product Owners devote less than fifty percent of their weekly time to the project, considering the poor quality of the software deployed by the supplier as the primary causes.

About the Supplier's Team:

This set of questions was designed to assess the performance of the software development supplier as well as their relationship with the ANI project team.

The responses to this set of questions indicate that Product Owners view the relationship and commitment of the software development supplier to the projects to be extremely valuable, and that they are always willing to provide clarity on any topic. However, the majority is dissatisfied with the supplier's performance. The lack of quality in software deployment is also a major concern for Product Owners, given that the software development provider does not do comprehensive testing. The Product Owners are also in agreement due to the fact that deliveries are consistently late.

About the Project Management Team:

This set of questions was designed to evaluate the performance of the project management team, as well as the evaluation of the tools used in the project management process and what might be done to improve the performance of the projects.

The Product Owners are pleased with the contribution that the Product Management team is making to all of the projects, specifically the incorporation of Agile principles and tools such as the KanBan Board, and risk management exercises, as it has had a significant impact on the projects, to know the current status of the user stories in development, and to provide general support when support is required to the supplier or the project itself. However, since the subject is relatively new and unfamiliar to the team, some believe that training on Agile concepts is necessary since they lack experience in particular areas.

Findings:

With the information drawn from this survey, it is possible to draw some conclusions and find opportunities for improvement to the IS Department as a whole.

From the analysis of the survey, it can be determined that the product owners have expressed a major concern over the time they have devoted to product owner obligations, indicating that the project is not proceeding as intended. There is also significant worry over the software development supplier, since teams are dissatisfied with the supplier's collaborative approach and are neither happy with its performance nor do they believe it has compromised with the project. A prevalent suspicion exists that the supplier does not release software with proper testing and rarely meets promised timelines. Concerning the project management team, the Product Owners believe that general training in project management techniques is required, demonstrating difficulty in understanding and using them.

Not only did the open-ended questions corroborate some of the reported flaws, but they also raised other worries. All participants verified a lack of available weekly time devoted to the project, making it the most significant source of barriers coupled with the poor quality given by the software development provider. This is attributed to the supplier's lack of testing, which causes extra issues for the organization, since it makes UAT not only more difficult but also more time-consuming. In addition, there was considerable attention on some matters defined by the product owners, including issues with the Product Owner's lack of awareness of their responsibilities, project management approaches, and the development of user stories and acceptance criteria.

In addition to the previously identified issues, product owners praised not only the IS Department but also the project management team for their efforts in regard to the organization of the work using agile methodologies, general support in the activities and tasks, and especially in the connection with the supplier, by assisting in the removal of some of the previously identified obstacles, as well as for their assistance with technical questions. It is then suggested that greater project scheduling assistance is required, as well as shorter development sprints for improved supervision of the development efforts by the supplier, and improved UAT efforts and allocation not only by the product owners, but also by the personnel that will use the final application on a daily basis. The mentioned difficulties fully include the experience encountered in ANI. Problems were recognized as a result of daily interactions with both the project teams and the product owners.

Regarding the department's compliance with the employed project management approaches, the experience is not uniform. There is a clear distinction between product owners, as some demonstrate naturalness with the subject matter, demonstrating correct usage of the KanBan board as well as correct description of user stories and acceptance criteria, whereas some Product Owners not only do not demonstrate the same abilities, but also have trouble accepting different methodologies from what they are accustomed to, raising issues for the project management team.

However, the second group of questions demonstrated that the most significant challenges were associated with the software development supplier. A collection of entropy parameters concerning communication and the quality of deployed software were revealed. Due to a lack of experience with

ANI's industry, the supplier shown a lack of comprehension in the user stories, needing developers that understand the business in question. Rarely did the software development team meet the established deadlines, necessitating the engagement of parties not directly involved in the project to resolve these issues. Consistently, the quality of the released software fell short of expectations, exposing existing concerns and previously unknown defects. As the pressure to meet project deadlines grew, the absence of software testing became increasingly apparent.

Regarding the same issue, the ANI project team's lack of availability may help to justify and resolve these problems. The lack of UAT revealed by the product owners is worrisome, revealing the absence of personnel available, a methodology and know-how, causing the UAT to be performed more arbitrarily, increasing the likelihood that some requirements are not adequately tested and that critical software bugs are not discovered, causing major problems in later stages of the project.

As a result, it is evident that some issues can be addressed at ANI's IS department, regarding project management methodologies and UAT, necessitating further research so that future improvements can be made to the department, which will be explored in the next Chapters.

2.4.2 Software Issues Analysis

There is a lack of documented information on UAT performed on software applications at ANI. For that reason, personnel from the IS department of ANI was asked about data containing information on software issues that could help to measure the extent of those problems. However, that information was either dispersed or did not exist for consultation.

Therefore, information was gathered from Product Owners, prior files, and the KanBan board, and an estimate of the percentage of user stories with recognized supplier problems was calculated. The outcomes of the investigation are shown in Table 1:

Table 1 - Software Development Issues

Project	Done US	US w/ Bugs	% US w/ Bugs
SIFIDE	45	7	16%
SGI	38	15	39%
SGP	19	9	47%
Orçamentos	13	-	-
Deslocações	2	-	-
Total	**117**	**31**	**26%**

As demonstrated by the study, the software developed by the software development supplier is of poor quality, as in the SIFIDE, SGI and SGP projects 16%, 39% and 47% of the user stories contained bugs. These issues were identified by ANI personnel, demonstrating that UAT is aiding with verifying that the software operates as intended.

On the other hand, due to the presence of bugs in the developed software, it may be argued that a well-executed UAT could assist to uncover other bugs that would prevent future trouble.

3. Research Background

In this chapter, some of the most crucial software development methodologies are uncovered and compared to find out the differences between them and how they contribute to the software development process.

3.1 Software Development Methodologies

3.1.1 Waterfall

The Waterfall model was initially presented by Royce in 1970 where it paradoxically defended the need for a more iterative approach to software development (Royce, 1970; Larman & Basili, 2003).

Royce describes what he considers to be the two necessary steps for all software development process: an analysis step and a coding step. He then proceeds to assert that any implementation consisting of only these two steps is doomed to fail. Some additional steps were proposed for developing a software program to a customer, which would become the foundation for the Waterfall model.

The multiple stages are executed consecutively, one after the other, and tasks can be split appropriately (Trivedi & Sharma, 2013), continuing to the next phase only after the previous one is complete (Bassil, 2012). Although the designations may vary depending on the author, as shown in Figure 6, Royce's Waterfall model can be simplified down to a five-step process.

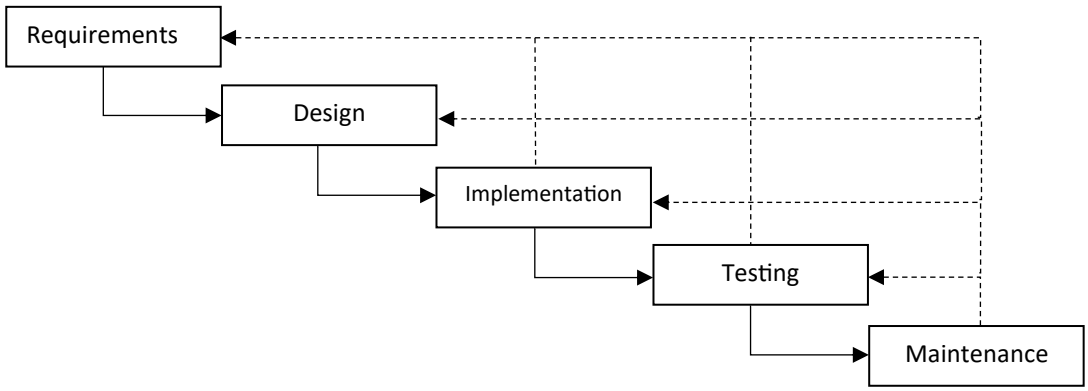

Figure 6 -Waterfall Model (Bassil, 2012)

Even though the waterfall model progresses in a linear fashion, each phase can be iterated on as many times as necessary to ensure that the project is finished (Bassil, 2012).

It focuses on a well-defined and exhaustive set of requirements, before starting the actual design and implementation of the project (Dima & Maassen, 2018), being the most suited for software projects where requirements are clearly defined (Saeed et al., 2019), and quality is given precedence over cost and schedule (Munassar & Govardhan, 2010), since quality evaluation occurs at the conclusion of each step before moving on (Petersen et al., 2009).

The waterfall model illustrates a more traditional approach, from the top to the bottom of the organization, with minimal or no contact between the client and the development team (Dima & Maassen, 2018).

3.1.2 Scrum

Scrum is an Agile management and control framework for software development created by Jeff Sutherland and Ken Schwaber that facilitates effective teamwork via an empirical, iterative, and incremental approach, intended for small teams of no more than 10 people (Schwaber & Beedle, 2002; Schwaber & Sutherland, 2020).

It can be utilized in a range of sectors, such as software development, running a company's division, or building a specific product, as it is not a predetermined set of processes but rather an adjustable framework based on the user's requirements (Gonçalves, 2018).

Standing on its three empirical pillars of transparency, inspection, and adaptation, it is intended that the process is transparent, that team members frequently inspect scrum artifacts and data working towards the sprint goal (McGreal & Jocham, 2018), and that teams can adjust the process to achieve the most advantageous business outcome.

The three branches of its structure are the scrum team, the scrum artifacts, and the scrum events. Three roles comprise the scrum team: the scrum master, the product owner, and the development team. The Scrum Artifacts are a collection of deliverables that enable the team to document and assess the product development process. This consists of a product backlog, the definition of a sprint backlog, and the completion of a release-ready increment. Also release burn down charts can be used, and a previously agreed-upon definition of done, so that everyone on the team understands what it means when an increment is marked as completed. Scrum events comprise the Sprint, Sprint Planning, the Daily Scrum, the Sprint Review, and the Sprint Retrospective. Each of these notions will be examined in further detail in the subsequent chapter.

One of Scrum's greatest obstacles occurs when the team is working for an external client. If the client lacks a clear vision of the product, the team will perform similarly, resulting in a product that does not meet expectations (Highsmith & Cockburn, 2001).

Scrumban and Scrum-XP are the most prevalent variants of the Scrum Framework, with Scrumban bringing the flexibility of Kanban's task board to work allocation (Petricioli & Fertalj, 2022), and Scrum-XP combining the project management skills of Scrum with the development capabilities of XP, therefore addressing their limitations (Mushtaq & Qureshi, 2012).

3.1.3 DevOps

The term "DevOps" combines the terms "developer" and "operations." Despite the lack of a universally accepted definition (Roche, 2013), numerous researchers have attempted to define the term.

According to Ebert et al. (2016), "DevOps" is a set of practices that enables the continuous delivery of software through automated development, releasing, and monitoring, made possible by improved communication between departments that perform multidisciplinary tasks as opposed to performing functions separately.

Smeds et al. (2015) examines the various existing definitions before defining DevOps as a collection of engineering process capabilities, including planning, development, testing, and release, made possible by a variety of technological and cultural factors.

de França et al. (2016) rejected DevOps as a methodology or a method, ultimately defining it as “a neologism representing a movement of ICT professionals addressing a different attitude regarding software delivery through the collaboration between software systems development and operation functions, based on a set of principles and practices.

Through a set of guiding principles of collaboration, automation, measurement, and monitoring (Lwakatare et al., 2015) and quality assurance (de França et al., 2016), it aims to achieve shared responsibility between dev and ops, continuous development of functionality, and the measurement of operational data (Lwakatare et al., 2015), in addition to other significant features that will be elucidated in further depth.

DevOps concepts cannot be directly opposed to agile principles, nor are they directly independent; nonetheless, they are complimentary. Lwakatare et al. (2016) asserts that agile concepts, as articulated in the Agile Manifesto (Beck et al., 2001), are necessary for the effective implementation of DevOps (Lwakatare et al., 2016). There are numerous issues with Agile development that can be resolved with the adoption of DevOps, including the delay of new features to customers, incompatibility between software components, and improper testing, that can be vastly enhanced by DevOps, specifically through automation and quality assurance (Almeida et al., 2022)

3.1.4 Product Management

Software Product Management is defined as the “discipline which governs a product from its inception to the market in order to generate biggest possible value to the business” (Ebert, 2007). Therefore, it assumes the premise that any software or digital product production can be managed as any other product, except for distribution costs, as those do not entail costs for the company (van de Weerd et al., 2006).

The Product Manager is the leading role in Product Management, the one responsible for the tasks related to the product, from the inception to the market (Ebert, 2014), having profound knowledge of the customer, data, business, market, and industry (Cagan, 2017). It must have influence on product decisions, authority, access to resources and influence on interaction and collaboration, providing a connection between top and lower-level management.

Identifying fragmentation in software product management knowledge, van de Weerd et al. (2006) developed the first framework detailing the areas related to product management, as well as all the stakeholders involved. It defines an iterative life cycle for the product focusing on portfolio management, product roadmapping, requirements management and release planning areas, based on observational studies. According to Kittlaus & Fricker (2017), while helpful, the framework does not include every task required of a product manager.

Several other frameworks were developed afterwards. Ebert & Brinkkemper (2014) provide a way to manage a software through its lifecycle through strategy, concept, market entry, development and evolution, emphasizing its continuous nature of the process; the Pragmatic Framework, a blueprint focused on product creation and marketing (*Product Management Framework* | *Pragmatic Institute*, n.d.); and the ISPMA SPM framework V.2.0, a framework that integrates several other methods, including the frameworks from Ebert and van de Weerd et all (Kittlaus & Fricker, 2017).

3.2 Comparison Between Software Development Methodologies

This Section will comprise a comparison between the four software development methodologies identified in the previous section – Waterfall, Scrum, DevOps and Product Management. There will be a comparison of Roles in Section 3.2.1, of Activities in Section 3.2.2, and of Deliverables in Section 3.2.3.

3.2.1 Roles

The success of an organization is dependent on the separation of roles within a team. A clear delineation of the roles reduces ambiguity and conflict and increases team collaboration since each team member

understands precisely how he is expected to behave within the team and what is expected of them (Zhu et al., 2006).

The previously described approaches place special emphasis on roles, since the right functioning of the team enables a more fluid and straightforward implementation of the concepts advocated by each of them. Therefore, it is essential to clarify these roles and their differences.

The Scrum framework has a clearly defined team structure, as in the Scrum Guide (Schwaber & Sutherland, 2020), constituted by the Product Owner, Scrum Master, and Development Team, as opposed to the other three cases that have roles that are dependent on various factors, such as the size of the team or the type of product. Therefore, the following comparison should be taken as a starting point to implement a particular methodology, and not as a rule of thumb.

The must-have roles for the Waterfall methodology (Trimble & Webster, 2013), Scrum Framework (Schwaber & Sutherland, 2020), DevOps (Bass et al., 2015), and Product Management (Cagan, 2017) are shown in Table 2, as well as the detailed duties reported for each one of them.

In the Waterfall methodology, the Project Manager is responsible to understand how to best implement a certain project and to ensure that the execution is followed as planned, as well as handling everything that concerns the team. In the Scrum framework, those responsibilities are split into the Product Owner and Scrum Master's duties. It should be noted that in the Scrum framework the teams are self-organized (Schwaber & Sutherland, 2020), therefore there is no need for the Scrum Master to coordinate project activities, working solely as support to the team and to the Scrum Framework. The roles of Service Owner and Team Lead in DevOps have similar duties to the Scrum ones, and that should not come as a surprise, since it is a common practice to integrate DevOps culture into the Scrum Framework (Stray et al., 2019), therefore having one Product Owner and Scrum Master taking the responsibility in that combination, being also common practice to identify DevOps Evangelists within the team, people who are familiar with DevOps culture and are able to influence and educate the rest of the team (Freeman, 2019). Regarding Product Management, the vision is different than in the previous mentioned approaches, therefore the Product Manager assumes a stronger business stance, being the major priority to bring to market a consumer approved product. For that reason, a Product Marketing manager works closely with him and the team, providing market and sales knowledge so that the product can be developed and placed according to customer needs.

The Software and User Experience Designer is a key role in the Waterfall methodology. It intervenes in the Design phase of the methodology, being responsible for translating the set requirements into Software Design, including everything from the system's architecture to the user interface. There is no defined Designer in the Scrum Framework team, thus the requirements and potential mock-ups should be ready before the beginning of the sprint, so that the developers can implement, the same happening with DevOps. In Product Management, the Product Designer works closely with the with the rest of the team, so that the coordination between the vision for the product and the actual solution is in accordance with what was projected.

Table 2 - Comparison Between Roles

Waterfall	Scrum	DevOps	Product Management
Project Manager · Report to senior management · Users' communication · Planning, scheduling, and coordinating project activities · Budget, schedule, risk, and quality control · People management **Software/UI Designer** · Software architecture design · Defining and understanding market segments/user to target, following requirements · Translating the understanding to the product concept · Iteratively refining the design **Quality Assurance Engineer** · Define testing approach, timeline, and related deliverables · Code verification with the quality assurance team · Report bugs or defects back to the development team	**Product Owner** · Maximizing the value of the product · Product Backlog management (accountable) · In charge of all the decisions regarding the product **Scrum Master** · Promoting and supporting the scrum framework · Monitoring team interactions · Make sure that the value produced by the team is maximized · Removing process and interaction bonuses · Ensuring timely and complete execution of scrum activities	**Service Owner** · Coordination with stakeholders and the team · Handles system-wide requirements and work prioritization · Elo between the team and client · Communicates the vision for the service **Team Lead** · Facilitating the team and obtaining resources · People management · Not responsible for technical activities such as planning and scheduling. **Quality Assurance Engineer** · Monitoring the service · Managing problems during the service execution · Performing analysis to solve the problems (using automated tools) · Finding causes and providing solutions for those problems. **Gatekeeper** · Decision to move a service to the next stage of deployment **DevOps Engineer** · Automating the development and deployment pipeline · Managing tools used in DevOps	**Product Manager** · Evaluating business opportunities · Determining what gets built on the product backlog · Lead all product activities, from the beginning until the market · Knowledge of the customer, data, business, market, and industry **Product Designer** · Collaborate with product manager and engineers/developers · Product discovery · User experience design, as opposed to user interface~ · Prototyping and user testing · Interaction and visual design handling **Product Marketing Manager** · Representing the market to the product team · Determining the positioning, messaging, and market plans · Engaging with the sales team · Keeping close contact with the Product Manager

For the success of a software product, its quality is of utmost importance. In the Waterfall methodology, when a phase is completed, a Quality Assurance Engineer verifies if the work done conforms with the company's quality standards, so that the finished work does not compromise the process. DevOps demands the same role to be fulfilled, corroborating two of the principles that support the DevOps culture: quality assurance (de França et al., 2016) and monitoring (Lwakatare et al., 2015). In Scrum and Product management cases, no such role is assigned. In Scrum, since it is considered that the

definition of done defined for a particular user story already assures that quality measures are taken, and the product already meets quality requirements (Schwaber & Sutherland, 2020). Instead, Product Management relies on automated tests (Cagan, 2017).

Although not present in Table 2 above, the common central elo in all approaches are the software developers, also referred as engineers. Their main duties include coding, testing, and deploying software, however there are differences on how their skills are applied. In the traditional waterfall method, the testing only starts after the coding is complete in one stance. On the other hand, in the Scrum framework the code is tested in every iteration (Leau et al., 2012). Also, the development team also actively participates in the planning for the next sprint. DevOps requires the same skills, adding requirements that facilitate the collaboration with Operations.

DevOps also involves additional required roles. The Gatekeeper oversees the deployment pipeline, permitting the service to advance to the next stage. There are, however, companies automating that process, as is the case with Netflix. The DevOps engineer supervises and manages the tools used in DevOps, as well as making major decisions regarding the team, and the automation in development and deployment pipelines.

Finally, there are many supporting roles that can take part in any of these, increasing the value of the team as a whole. A dedicated requirements engineer can be included in the Waterfall methodology, network technicians to assist in cloud automation in DevOps (Shah et al., 2018), or Data Analysts in Product Management so that the best set of data metrics is used, and the results are correctly interpreted.

3.2.2 Activities

Activities are the collection of elements that must be performed in order to execute a project. Each of the four approaches entails a distinct set of necessary tasks, each with its own distinct characteristics. Each of the four approaches entails a distinct set of necessary tasks, each with its own distinct characteristics and differences. These are presented in Table 3, along with their respective descriptions.

As previously stated, DevOps activities, or capabilities, cannot be directly opposed to the other approaches, since they are meant as a movement, or culture, considered to be essential in any software development team. Thus, in this section, Waterfall, Scrum and Project Management will be compared, finishing on how they relate with DevOps principles.

The Waterfall model starts by establishing the requirements for the software and organizing them in user cases (Bassil, 2012; Royce, 1970). There is not a dedicated activity for requirements in Scrum (Schwaber & Sutherland, 2020)- the requirements duties are shared by the various stakeholders in the project, team members, customers, and others, and are subsequently defined by the Product Owner in the Product Backlog. Requirements in Product Management is done slightly differently – evidence is

collected through a process of equilibrium between customers, executives and the product team. Thus, a Process Framing phase is initiated, that includes defining purpose and alignment of the team, as well as the risks involved, followed by a discovery planning by the team, to figure out solutions by using customer-oriented techniques. The team proceeds to generate product ideas to tackle the business problems identified previously.

Table 3 - Comparison Between Activities

Waterfall	Scrum	DevOps	Product Management
Requirements · Establishing the requirements for the software · Redefinition based on customers' needs · Definition through user stories or cases **Design** • Planning system's architecture • Systems documentation • Definition of database schema, diagrams, and user interface **Implementation** • Enactment of requirements and design • Written code **Testing** • Verification of the implementation • Code debugging • Software quality assurance for deployment **Maintenance** • After deployment fixes • Performance and quality enhanced • Software refinement • Customer reports and assistance	**Sprint** • Time-constrained event to release an increment • Usual duration between 2-4 weeks • Frames every other Scrum event **Sprint Planning** • Planning event to define what is going to be delivered in the sprint • Placing at the end of the Sprint to define what is going to be delivered in the next one **Daily Scrum** • Held at the beginning of every day of the sprint • Review of what was accomplished since the previous sprint • Definition of what can be done to achieve the sprint goal • Team identification of impediments **Sprint Review** • Held at the end of the sprint • Assessment of the increment • Product backlog evaluation – revised product backlog **Sprint Retrospective** • Meeting that concludes the sprint • Assessment of the previous sprint regarding people, relationships, processes, and tools • Identifying and planning potential improvements	**Continuous Integration** • Code integrated as soon as possible and shared with the team • Observe code long-term behaviour • Automated deployment integration **Continuous Deployment** • Deployment of changes after right after the testing process • Small continuous releases rather than large releases **Continuous Monitoring** • Data collection • Key metrics identification • Fast response to undesired behaviour **Continuous Testing** • Immediate feedback through early and automated testing • Allows prompt error correction **Feedback Loops between Dev and Ops** • Tight integration and collaboration between development and operations • Facilitated knowledge and feedback sharing **Infrastructure as Code** • Enforce consistency throughout software development • Infrastructure assembled as any part of the application • Easily reversible infrastructure builds to fix issues	**Product Discovery** • Discovery Framing: Confirming team purpose and alignment, as well as risks involved • Discovery Planning: Figuring out product solutions and planning discovery efforts • Discovery Idealization: Generating ideas • Discovery prototyping: Building feasibility, user, live-data, and hybrid prototypes • Discovery Testing: Assessing value and usability with the customers by recruiting users to test **Transformation** • Allowing the organization to adapt to new circumstances **Process at Scale** • Managing stakeholders' expectations • Share product knowledge • Consistent innovation

The paradox is as follows: it is common for the product team to be handled predefined requirements from the large customers or stakeholders, therefore, the product team is not given large possibilities of innovation. That is comparable to the other approaches, although it is considered that it rarely equals the quality of ideas generated by the team, as those are originated by interacting with users and customers (Cagan, 2017).

The Design phase in the Waterfall methodology starts only after the full conclusion of the Requirements phase, and is characterized by the planning and definition of the software architecture and user interface, being done prior to the implementation. Scrum and Product Management make no reference to design – Scrum, which supports the concept of self-organizing teams, may tackle this problem in a variety of ways, such as by having a parallel scrum team manage the whole design process (Hron & Obwegeser, 2018).

Product Management handles user experience and interface by combining the user prototyping and user testing phases. The prototypes are intended to help the team understand something at a cheaper cost and in less time. The prototypes can answer feasibility, user, and data concerns by constructing a tiny sample of the product and hiring chosen consumers to test it, therefore assisting the team in identifying the best ideas for the product it is attempting to develop. Considering that, a concept know as Minimum Viable Product (MVP) may be introduced. The MVP is an early version of the product with sufficient functionality to test the product and verify the concept early in the process (Ries, 2011). Although it is a notion related with Product Management, can also be merged with other approaches.

The three remaining Waterfall steps of implementation, testing and maintenance happens in successive phases, and are directly addressed through the Scrum framework. The major time unit in Scrum, the Sprint, defines the time that the Scrum team must release an Increment. Sprint Planning precedes a Sprint and intends to plan what will be Increment and how it will be achieved, and the Sprint Review evaluates what was achieved both by the Scrum team and the stakeholders involved. The Sprint is also supported by the Daily Scrum, a brief meeting at the beginning of every day that intends to review the work done and remove impediments in order to achieve the expected increment and remove impediments; and a Sprint Retrospective that closes the Sprint to plan potential improvements regarding people, relationships, processes and tools. Thus, Scrum focuses on iteratively delivering a useable increment at the end of each sprint by reducing the team's barriers and promoting internal cooperation (Bhavsar et al., 2020).

In contrast, Software Product Management makes no reference to implementation/coding, testing the code or maintenance (Kittlaus & Fricker, 2017). Instead, it focuses on giving the resources to change into empowered and accountable product teams whose performance is judged by business results, as opposed to being output oriented, and approaches for scaling the innovation process from small to big teams.

DevOps, being a set of capabilities that focuses on pipeline optimization, is not a software development methodology as the ones previously described. Although DevOps also includes implementation, testing and deployment, none of the other approaches adds a layer of constant operations adds a layer of

continuous operations (Mohammad, 2017), addressing the challenges that arise after the software is deployed. The implementation of DevOps is related with Agile methods, as it can be built upon its practices, especially continuous integration, as well its existing roles (Lwakatare et al., 2016). Thus, DevOps can be is commonly integrated with agile frameworks such as Scrum, although it does not apply to a specific one (Mohammad, 2017). By combining Agile and DevOps practices and capabilities exhibited in Table 3, a firm may realize significant project benefits, complementing one another and contributing to the team's progress (Almeida et al., 2022).

3.2.3 Deliverables

There is a variety of documentation associated with the various software development approaches that are being discussed. Therefore, a gathering of those was conducted regarding the Waterfall methodology (Kramer, 2018), the Scrum Framework (Schwaber & Sutherland, 2020), DevOps (Best, 2017) and Product Management (Cagan, 2017), and it is presented in Table 4.

The documentation concerning the waterfall methodology is organized in regard to the phase of development associated with it. As for the remaining ones, are listed along with a brief description on the characteristics and what it is intended to be accomplished.

The Waterfall Model is characterized by a big amount of documentation, present in every step of development, also including specific Deployment documentation, recording every step of the process that was conducted so that there are no measures that are taken without careful planning. That contributes on the characteristics of the Waterfall methodology – the emphasis on quality control of the product but also a big portion of time dedicated to bureaucracy.

That contrasts immensely with the other approaches. The Scrum Framework, as part of the Agile Movement and following the Agile Manifesto’s principles values “Working software over comprehensive documentation” (Beck et al., 2001), provides different purpose deliverables. Those deliverables, named Artifacts in the framework support transparency and focus, being a measure of progress.

The Product Backlog acts as replacement for the Requirements documentation in the Waterfall methodology, containing every requirement for the product to be delivered as intended, being an iterative Artifact, as details are being constantly added in order to add value to the product as long as it progresses. At the beginning of each sprint, in the Sprint Planning event, the team selects the items from the Product Backlog that it deems necessary to accomplish the defined Sprint goal, acting as a plan for the developers to follow iteratively. During the sprint, as the Sprint Backlog items are being completed, it reaches an Increment – a tangible step towards the Product Goal. That increment must follow quality measures defined by the team in it definition of done, allowing the team to concretely understand what are is required for the Increment to be considered.

Table 4 - Comparison Between Deliverables

Waterfall	Scrum	DevOps	Product Management
Requirements · RUD – Requirements understanding document · Functional and business requirement systems development **Design** • HLD- High-Level design document • LLD – Low-Level design document **Implementation** • Application • Unit Test Cases • Results from the developers **Testing** • Test Cases • Test Reports • Defect Reports • Issues Log **Deployment** • User Manual •Environment definition/specification • Issues Log **Maintenance** • User Manual • List of resolved production tickets • List of new features implemented	**Product Backlog** • List of what is needed to improve the product • Only source of work by the team • Product Owner is accountable **Sprint Backlog** • Sprint Goal • Selection of items from the Product Backlog to be achieved in the Sprint • How to deliver the increment **Increment** • Constitutes the product goal • Increments are additive to all prior increments • Must be usable	**Funnel** • Gathering of ideas in a funnel visual representation • Contributes to the maturity of ideas • Prior to the roadmap **Roadmap** • Path of product • Born from the vision and goals for a product **Release Planning** • Prior to the Product Backlog • Requirements for the forthcoming deployment • Contains every epic to be detailed	**Product Vision** • Description of the future that is trying to be created • Can take many forms • Provide the team with inspiration **Product Strategy** • Product or version planning towards the product vision • Product and market fit **Other Deliverables** • Documentation originated from used techniques

As already mentioned, DevOps adds another layer to software development practices, focusing on a continuous integration between developers and operations. Rong et al. states that there is a lack of documentation research regarding DevOps, that also being not very well defined due to being recently

adopted by companies, defending an automated approach for documentation in DevOps, defending the usage an easily accessible cloud solution to that end (Rong et al., 2019).

On the other hand, Best (2017) defends the usage of three vital deliverables to ensure the success of DevOps. The process starts with a set of ideas gathered in a funnel-like analogy. As the ideas mature, they descend into the funnel until the final concept is defined for the product. The pact of the product is then defined by the usage of a product roadmap, implicitly having the vision and goal of the product in mind.

A Release Planning document is also led afterwards, containing all the epics ready to be detailed for the product backlog.

Needless to refer that there is more documentation that can be associated with DevOps. However, that can be associated with the integration with other approaches. The Product Backlog, Sprint Backlog, and Increment are additional deliverables that may be derived from the common Agile integration, notably with the Scrum Framework.

There are no deliverables associated with the actual software development process itself in Product Management. It recognizes the need for a product vision, with a long-term view of what is being developed, and a product strategy outlining the products or versions required to achieve the vision, through a succession of product market fits, since the product and the business are the top priorities. The product vision has no predetermined structure and serves merely as a compelling element. Other deliverables may also be obtained by techniques employed in the different product discovery activities (for instance, prototypes); however, this is not required and depends on the product circumstances.

Cagan (2017) disregards the inclusion of product roadmaps in Product Management, arguing that they frequently result in poor business outcomes because many product ideas are doomed to fail, wasting time and resources; and by using a documented roadmap, the team will be led to view it as a commitment rather than a potential course of action.to view it as a commitment rather than a potential course of action.

In Table 5, a brief comparison of techniques regarding documentation/deliverables is provided.

Table 5 - Deliverable Comparison Summary

Software Development Approach	Quantity	Type of Documentation
Waterfall	High	Descriptive
Scrum	Low	End-User focused
DevOps	Medium	End-User and Pipeline focused
Product Management	Low	Product Goal focused

4. UAT Literature Review

In this Chapter, a Literature Review on manual UAT will be conducted. In Section 4.1, software testing will be briefly introduced, followed by the Literature Review Protocol in Section 4.2. The Results from the Literature Review will be presented in Section 4.3.

4.1 Introduction

Software testing is the process of verifying that the software's requirements and components meet expectations, with the goal of detecting defects. assuming a key importance in the successful completion of a Software Development Life Cycle, being considered the most crucial one (Hooda & Singh Chhillar, 2015). It is a complex task, as software does not behave like any other system, having an infinite set of ways that can fail. Testing the full extent of the software would be infeasible, due to its complexity, thus a process must be conducted that includes designing and executing those tests, so that problems can be identified and consequently solved accordingly (Tuteja & Dubey, 2012).

Depending on what is meant to be analysed, testing can be performed manually or automatically, contain many phases and types, and be tailored to meet the demands of the client.

The process starts by analysing the requirements, followed by test planning and preparation. After all is ready, the testing is executed according to plan and all bugs are reported and consequently fixed. Finally, test closure is conducted, where the personnel accountable for testing procedures verify the test artifacts and, if everything is according to quality assurance parameters, the software is released.

This accounts for the general procedure for testing, however there are various types – and sub-types – of software testing. Testing operations must be meticulously organized, as time is limited, and the process is lengthy.

Functional testing verifies that software corresponds to its functionality and behaviour specifications. Regarding that, there are various subtypes of testing:

- **Unit testing** is conducted by the developer and focuses on lower level issues;
- **Integration testing** examines the interoperability of software modules;
- **System testing** ensures that the integrated system operates in accordance with the defined requirements;
- **Acceptance testing** ensures that the program performs as intended by the customers, providing input to the software development team for potential changes;
- **White and Black box** testing checks code redundancy, and certifies that inputs and outputs of an application are correct, respectively.

Besides functional testing, non-functional Performance and Security testing must also be carried out. Using time, load, stress, and success parameters, performance testing verifies software functionality under various conditions. Security testing is to guarantee that the system's security is not compromised in any way and that there are no exploitable flaws (Hooda & Singh Chhillar, 2015).

The acceptance testing process being done in the right way greatly contributes for the whole functioning of the firm. This issue must be addressed by developing a simple, yet effective methodology, so that employees can test the entirety of the software in relation to the requirements previously defined by the Product Owner in the form of user stories, regardless of their knowledge of the platform and IS in general. This improves not only the detection and resolution of problems, but also their proper identification and communication with the software supplier so that it can be rapidly resolved.

4.2 Literature Review Protocol

The following research sought to describe or detail manual acceptance testing procedures or frameworks that may be used in a corporation utilizing Agile approaches, notably Scrum, or agile concepts applied to acceptance testing.

It was conducted using the Google Scholar Academic (scholar.google.com), a search engine that compiles the most notable sources of academic articles, thesis, and books.

The following search string was defined as it was considered it could deliver the best possible results:

- (“Acceptance testing” OR “Acceptance Test”) AND (Agile OR Scrum)

The search found 14200 results. The following requirements were established for the suitability of a certain result:

- Results must be dated 2008 or after.
- Research must be cited; only publications within the past three years were approved with a low number of citations.
- The source of the result cannot be from a website, a book, or a thesis.
- Only research in English, Portuguese or Spanish was considered.
- The primary focus of study must be a manual acceptance testing procedure or technique that adheres to Agile or Scrum principles; hence, completely automated approaches not considered.

Relevance was utilized to sort Google Scholar results. The process of screening was subsequently undertaken through the individual reading of each article's abstract. Depending on the abstract's substance, the papers were thoroughly examined and then selected or rejected based on the research

context. When no relevant papers were obtained for 30 consecutive articles, the research was terminated.

Ten papers were deemed relevant to the study. Subsequently, a forward snowballing approach using the listed citations (Felizardo et al., 2016), as well as backward snowballing approach by means of references (Wohlin, 2014) was employed to these publications to identify any missing relevant research on the issue, both using the Google Scholar search engine. Although snowballing does not substitute database searches, it allows to widen a literature review, since by starting with relevant papers it is possible to check publications not only referenced by the authors, but also newer studies that used their findings to provide a new approach or a continuance on a certain subject. In both of these approaches, publications were sorted by title, abstract, and subsequently by reading the full article, selecting five publications by forward snowballing and two by backward snowballing.

It is also essential to note that the explicit inclusion of the article in Agile or Scrum principles was not regarded obligatory in either technique.

The results of the research are summarised in Table 6:

Table 6 - Literature Research Summary

Research Layer	**Number of Results/Articles**
String Search results	14200
Abstract Filtering	75
Full Article Filtering	26
Selected Articles	**10**
Available Articles for Forward Snowballing	215
Selected Forward Snowballing Articles	**5**
Available Articles for Backward Snowballing	207
Abstract Filtering	33
Full Article Filtering	5
Selected Backward Snowballing Articles	**2**
TOTAL SELECTED ARTICLES	**17**

4.3 Results

The bulk of relevant papers had agile-related connections and principles. Publications proposed a framework-like approach to agile (Al-Hurmuzi et al., 2018; Hongying & Cheng, 2011; Hu et al., 2021; Otaduy & Diaz, 2017; Pillai & Hemamalini, 2022), or particularly to the Scrum framework (Aamir & Khan, 2017; Elallaoui et al., 2015; Geras, 2008; Löffler et al., 2010; Pandit & Tahiliani, 2015). On the other hand, an agile principles-based framework was established (Harridon et al., 2021; Padmini et al., 2016; Pillai et al., 2019; Rajasekaran & Nithyarao, 2017; Tonkin et al., 2020) present a framework for UAT with no agile association. (Long, 2020) and (Sanjaya & Andry, 2021) have no associated approach, setting valuable principles to acceptance testing that can be used in a variety of circumstances.

Pillai et al. (2019) advocate including UAT at each stage of the testing development life cycle. Using Agile principles, a set of activities is developed to prevent confusion during UAT and assist teams in deciding how to perform the process. It involves the identification of responsibilities within the testing team, the modes of testing that will be employed, the testing timeframe, and the documentation tracking requirements, with the claim that it decreases the occurrence of defects and project expenses.

Based on the re-use of test cases, Hu et al. (2021) provide a strategy for optimizing UAT in projects with an existing process. The use cases are optimized by dividing them into two levels: test suites and optimized test cases. The test cases in the testing suite have previously been sorted and reflect various testing situations. Therefore, use cases are selected based on requirements, and new testing suites are constructed by mapping requirements using the use case library. Finally, the test cases inside the test suite should be ordered according to priority parameters, scene significance, and execution time. For test suite prioritization, a function "importance of requirement r" V(r) is defined as

$$V(r) = w_1 V(u) + w_2 V(s),\ w1 + w2 = 1$$

, with V(u) indicating the significance of user choice and V(s) representing the importance of software nature, and w1 and w2 being their weight ratios. The higher a requirement is in the list of priorities, the greater its importance.

Padmini et al. (2016) provide a UAT framework, centred on team-wide transformation and incrementally enhancing the process. It is recommended to select domain experts as opposed to technology experts and giving training as needed for testing and writing test scenarios. After that, defends a thorough Scrum-like testing framework that comprises release prioritization, separated into modules, from which testing scenarios will be picked to be performed in an iterative manner by the many testing teams allocated to the project. The comprehensive framework includes Scrum-inspired objects and events, the organization of test cases in a scenario backlog, scenario planning sessions, and a sprint-based timeframe similar to the software development Scrum methodology.

Pillai & Hemamalini (2022) eliminate the need for automation by introducing a hybrid UAT technique that partially mixes Agile and Traditional concepts and incorporates team members from several departments in its design. UAT is conducted at each stage of the software development life cycle, and

consequently after each testing phase. This entails the introduction of a "test case document initiation" for the conversion of requirements documents to test cases, detailing document history and reference documents to be consulted, that should be updated as long as the tests are being performed, contributing to reduce the time spent by the tester and getting user approval. The reuse of current and previous test cases is also encouraged, proposing a sample test case template for test case execution. Lastly, it is asserted that by using the hybrid model, the number of found errors increase, costs decrease, and thus, software quality improves.

Hongying & Cheng (2011) provide a generic Agile quality assurance procedure suitable for small to medium-sized teams, which is adaptable based on team size and maturity level. Although UAT is not the topic of this work, the whole procedure may be used to this testing step. After the testing phase is complete, the model focuses on peer review, defining key process areas and giving guidelines, benefits, procedures, best practices, templates, customization options, and maturity levels for each one. The roles in the review are established based on the model, and the peer review aims to detect the issues identified during testing and convey them effectively; if the flaws are validated during the review session, a template is completed. The method identifies four maturity levels for applying the model, indicating the extent of compromise or familiarity of the team with the model.

Several writers argue for a distinct technique, involving the creation of sequence diagrams that represent user stories in the Scrum methodology. Löffler et al. (2010) propose the systemization of test design by converting requirements – converted in a user story - to a UML language model. Those models intend to simplify a user story description by the users, which will be afterwards linked to a sequence diagram. With developers and testers working in parallel, it is implemented by using a selected fixture, performing automated UAT. Each sequence diagram is coupled with a testing table, ensuring that UAT is conducted on all user stories. Elallaoui et al. (2015) recommends the creation of automatically generated UML diagrams, by the inclusion of an algorithm in the Scrum framework.

With help from automation – although that can be done manually- information is extracted from user stories to build said sequence diagrams and used to easily generate test cases. Comparably to Elallaoui et al. (2015) and Löffler et al. (2010), Pandit & Tahiliani (2015) propose a framework for translating user cases/stories and acceptance criteria into natural language UAT, suitable for general agile methodologies. Natural language “intends to make computers understand the statements or words written in human languages” (Chopra et al., 2013).

With that end, information is extracted from user stories and acceptance criteria using a predefined template and and translated to document object model in XML is built. That will be the basis used for building positive, negative, and non-functional user acceptance tests. Finally, after that iterative process, those can be not only easily filtered for analysis purposes, but also requirements are associated with features, roles, acceptance criteria and user acceptance tests, automatically generated in an Excel sheet to provide traceability.

In contrast to a tabular traceability matrix, Rajasekaran & Nithyarao (2017) give a novel viewpoint on diagram usage in UAT by presenting a framework that implements a traceability matrix by deriving from

the notion of mind maps. Test groups are formed, and nodes are added, then linked to requirements or use cases. Every requirement is paired with a test case, giving in a simplified perspective of the whole testing process, which claims to increase test analysis and coverage.

Otaduy & Diaz (2017) identify lack of time, lack of motivation and lack of knowledge as the main root causes for the lack of customer involvement in UAT. A methodology is developed to solve those recognized problems by proposing the usage of a wiki page to facilitate collaboration between users, containing information about sprints, software features and a testing page where users performing tests share results of testing, complemented with commentary. Test cases are defined with a tool named test map, an adaptation of the concept of test maps, offering a simple, yet common notation for its users. The users are then responsible for defining a UAT case, providing a data set and confirming expectations – thus, inputs and outputs. The test cases are described under three variables – customer, organization and web application under test. The methodology also tries to provide means to complement in-person meetings with asynchronous ways for customers to collaborate, conducting UAT on their own, being characterized as a "self-paced process".

These approaches greatly contribute to a traceability component of UAT, being the "ability to describe and follow the life of software artifacts", by connecting related ones. In software testing it can help teams to locate faulty software, by locating issues in software, avoiding redundancy, and analysing changes (Parizi et al., 2014). In addition, Elallaoui et al. (2015), Löffler et al. (2010) and Otaduy & Diaz (2017) use of automation, which plays a part in the technique, unlike the other papers in this research.

As previously mentioned, in addition to Elallaoui et al. (2015) and Löffler et al. (2010), Aamir & Khan (2017), Geras (2008) and Pandit & Tahiliani (2015) are also focused on Scrum, proposing the incorporation of a testing backlog in the existing framework, containing test cases to be processed. Aamir & Khan (2017) suggest that the testing activities start when a sprint starts with the testing team working in parallel with the development team, thus inducing the creation of an "enhanced scrum framework". The test cases should be prepared based on functional requirements translated to user stories related to the current sprint.

Following this, a process of prioritizing sorts the test cases according to risks, functionality, and frequency of defects, presents a clear description of the roles that should be involved in every quality assurance activity, and provides a tool to evaluate software quality based on criteria. Therefore, despite the fact that the essay does not solely focus on UAT, it contains useful advice for its execution. Al-Hurmuzi et al. (2018) suggest a manual method to UAT based on the clear definition of roles and responsibilities of business users, so that the quality assurance team can accurately assign test cases. It also includes which documentation to complete while performing test cases and providing feedback, all utilizing an Excel spreadsheet as a database. Geras (2008) presents the notion of test target, a list of items to be tested in a test backlog, including concrete individual testing strings and scenarios, arguing that communication may be enhanced since feedback can be obtained more quickly.

In conjugation, a set of practices are outlined so that testers can conduct tests more efficiently, followed by a guided exploratory testing process that was conducted by adhering to Agile principles by prioritizing

the backlog, placing an emphasis on skilled and motivated individuals, and creating documentation that is both simplified and easily accessible, controlling the progress with a burn-down chart for increased visibility.

Long (2020) states best practices that contribute for UAT to be more human centric, avoiding errors that can be avoided through communication barriers or "environmental frustrations" by providing a template that contributes to the increase of those values from the beginning to the end of the testing process, but does not provide a clear framework or methodology. Similarly, Sanjaya & Andry (2021) conducted a study utilizing ISO 9126 and its characteristics to implement and evaluate applications with a focus on functionality, outlining questions for the testing team to ask regarding the suitability, accuracy, interoperability, and security of the software in question to ensure that software quality is not compromised. ISO 9126 is an international standard for assessing software quality. It does so by considering both internal and external software qualities, as well as their relationships and attributes, which are organized hierarchically as a tree of characteristics and sub-characteristics (Fahmy et al., 2012).

Last but not least, Harridon et al. (2021) and Tonkin et al. (2020) offer a unique viewpoint on UAT by merging testing of software and hardware components. Harridon et al. (2021) give a technique for testing a Static Flight Simulator and identifying which components must be tested, offering a perspective on UAT applied to a huge project. The testing is divided into the software portion and the physical portion, and then compared to the actual airplane, with the output being the components to be tested. The simulator has been divided into several entities, and specialists have been entrusted with distinguishing between them. If a variance was observed, calibration would be required.

Tonkin et al. (2020) created a technique for UAT of wearables by incorporating a comparison component between the first and second versions of the wearable to see if the modifications are significant enough to be embraced by its users or customers. Participants acquired data by doing daily tasks in order to test the performance of the wearable in terms of its components and how those components work under different situations, and to determine if the wearable gathers the required information to discriminate across activities. This evidence-based method was found to be valid for comparing successive generations of a hardware and software product and addressing the issues that arose, given that testing is conducted at various stages using a meticulous data analysis and comparison in terms of stability, reliability, and activity recognition.

Due to the large number of concepts discussed in the papers, Table 7 gives an overview of the approaches, characteristics, and artifacts utilized in each of the previously selected publications.

Table 7 - Approach and Characteristics features of UAT Literature

	Approach		Characteristics														
	AG	SC	FW	CO	US	TC	RL	TS	RQ	EV	TD	TF	TR	NL	TB	DL	AT
(Aamir & Khan, 2017)	X	X	X	X		X	X					X			X		
(Al-Hurmuzi et al., 2018)	X		X			X	X		X				X			X	
(Elallaoui et al., 2015)	X	X			X	X					X		X				X
(Geras, 2008)	X	X	X	X						X					X	X	
(Löffler et al., 2010)	X	X	X		X				X		X		X				X
(Otaduy & Diaz, 2017)	X		X	X		X		X			X		X			X	X
(Padmini et al., 2016)	X	X	X	X			X	X	X			X			X	X	
(Pandit & Tahiliani, 2015)	X	X	X		X	X			X		X		X	X			
(Hu et al., 2021)	X		X			X		X	X			X					
(Pillai et al., 2019)	X	X	X	X			X			X		X				X	
Harridon et al. (2021)			X						X							X	
(Long, 2020)			X	X						X						X	
(Pillai & Hemamalini, 2022)	X		X			X			X			X					
(Rajasekaran & Nithyarao, 2017)			X		X	X			X		X		X				
(Sanjaya & Andry, 2021)						X				X							
(Hongying & Cheng, 2011)	X		X	X		X	X			X						X	
(Tonkin et al., 2020)			X	X					X			X					

Agile (AG); Scrum (SC); Framework (FW); Communication (CO); User Stories (US); Test Cases (TC); Roles (RL); Test Scenarios (TS); Requirements (RQ); Environment (EV); Testing Diagram (TD); Time Framed (TF); Traceability (TR); Natural Language (NL); Testing Backlog (TB); Deliverables (DL); Automation (AT)

It is worth noting that the term "Framework" is intended to embrace frameworks, methods, approaches, or anything that has an ordered sequence of stages, whereas the term "testing diagram" encompasses any diagram-like procedures, independent of type or language.

Most publications represent acceptance testing frameworks or methodologies that intend to accelerate and optimize the overall process, providing foundations that can be used in UAT. There is a clear focus from the publications on test cases, communication among testing teams, deliverables and requirements, those concepts being widely mentioned on the selected publications.

Although there is an apparent large number of publications, there is a dearth of literature on manual UAT, particularly on manual techniques of conducting it. Even if it's just partially automated, many methods incorporate some form of automation.

Most publications give either a generic framework or a set of methods that UAT teams may follow to be more efficient and successful in defect identification, but they are not specialized to a particular case. When placed on a separate project or conjugated with a framework, they are not anticipated to perform at their maximum capability. Moreover, although the writers may believe that the offered papers describe a straightforward procedure, some of the concepts are quite complicated.

Despite this, regardless of the publication type, all selected publications contain a variety of concepts, principles, activities, and documentation that may be translated or modified to a software development project. In light of this, it was determined the need for a framework that is both effective and easy, concentrated on the business capabilities of its users, being able to follow without having any business or development expertise.

As previously identified, business users at ANI exhibit signs of resistance to change and a lack of a UAT methodology; therefore, the process that will be developed will challenge these concerns, being adaptable from small to medium-sized software development projects with concepts from the Literature Review.

5. Research Methodology

The Literature Review in Chapter 4 covered many methodologies used for UAT in the Agile methodology, highlighting essential concepts and approaches that may be implemented to ensure the process works as effectively and smoothly as possible. This chapter will discuss the research methodology employed for the process proposal suited for ANI, including the research approach, data collection, and considerations.

The research methodology follows an iterative strategy that aims to encompass insights from the previous chapter, accumulated experience with UAT at ANI during the internship, and input from prospective users and testers.

In order to create a model capable of optimizing the firm's process, a first iteration will be done by merging the corporate context collected in Chapter 2 with the concepts gathered from the Literature Review. Through the identification of process opportunities, the objective is to provide a process that C can aid ANI in UAT.

The process resulting from the first iteration is applied to one of the various projects that ANI has, with the model being deconstructed as much as possible by finding improvements and correcting flaws, taking into account the inputs from the project's Product Owner that is also the developer and tester of the project, in order to produce the second iteration of the proposal.

The second iteration is evaluated by doing UAT on a second, more complicated project application. As the process is put to the test against a different project, it is intended to be further validated so that it becomes more mature, and biases are reduced. Interviews with pertinent ANI´s professionals will result in the model's third and final iteration. The procedure is depicted in Figure 7.

Between the first and second iteration, the data collection process is conducted through a series of meetings, allowing participants to share their insights in a more informal manner by sharing their experiences and allowing their opinions, perspectives, and experiences to flow naturally. UAT was conducted on the application that corresponds to the DIPI project. It is the most straightforward application in ANI's portfolio; a website that allows the user to browse, update, and extract information about the current Perin Network projects, with its development being conducted "in-house" and without a large number of user stories that were accounted for.

Between the second and third iteration, data was collected via a presentation of the second iteration's process, followed by an interview with relevant personnel from the company, focusing on pertinent predefined points that significantly contributed to the process's validation. In order to further validate the UAT process, it was defined that performing UAT in one project was not enough to justify its usage for ANI’s project applications. Therefore, a larger, more complex project was selected, that being the SGP Project, or Expert Management System.

The Expert Management System is a project consisting of an application that provides tools for ANI's technicians to select an Expert in a designated field to evaluate if a project is worth of funding when they do not possess the capabilities to do it themselves.

All participant data was gathered with their agreement, and they were informed of the study procedures and objectives both in writing and during planned sessions.

Finally, a final iteration of the UAT process will be proposed, followed by a discussion of quantitative results from both UAT Applications.

The research methodology is summarised in Figure 7.

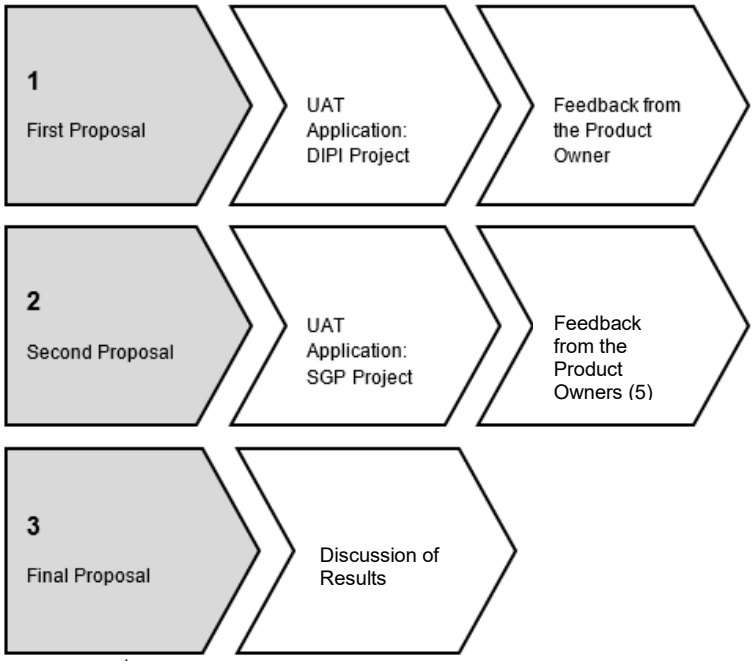

Figure 7 - Research Methodology

6. First Proposal

In this chapter, a First UAT Process is proposed in Section 6.1, followed by its UAT application in ANI's DIPI Project. In addition to getting feedback from the Product Owner, conclusions will be drawn from the UAT experience in Section 6.2.

6.1 Literature-Based First Proposal

As previously described, Agile and Scrum teams may utilize a variety of strategies for UAT. These processes contain extremely valuable concepts that may be used by any firm; they are either specialized for highly particular instances or feature broad procedures that are recommended to be relevant to all cases.

As indicated, the software development process at ANI comprises of a hybrid Scrum method, supplemented using a KanBan board for user story structure and prioritization, with the development team being an external provider. Although Scrum is one of the most popular software development methodologies, it is not specific regarding software testing; hence testing must be tailored to the methodology and the company's values, principles, and people.

In this chapter, a UAT procedure will be proposed for ANI, which will be adapted to the Agency's needs and backed by the results of the literature review completed in Section 4.3.

Two of the three roles identified in the scrum methodology are redefined by the proposed process:

- The Product Owner is accountable for the UAT Process.
- The Scrum Master is additionally responsible for inspiring the team to do UAT.
- The addition of a Tester to the Scrum Team. The testers are firm employees with business expertise in a particular industry who are equipped to conduct UAT. Testers are assigned by the Product Owner.

The proposal comprises of a two-phased procedure:

- The first phase, **UAT Planning**, conducted just once every Sprint by the product owner.
- The second part, **UAT Execution**, is conducted by the Tester.

As acknowledged in Chapter 2, ANI has a deficiency of knowledge regarding UAT practices; therefore, it is imperative that the personnel involved in the process understand it, so that the entire team is on the same page and working toward the same objective. The team must have the skills necessary to follow the process and be able to follow UAT steps practically (Al-Hurmuzi et al., 2018; Hongying & Cheng, 2011). Padmini et al. (2016) state that feedback sessions should be held, not only to ensure that Agile

techniques are implemented appropriately, but also to ensure that testers recognize their value in the UAT process. Also, it is noted that familiarity with the testing process greatly contributes to personnel's improved ability to manage software tools (Harridon et al., 2021; Otaduy & Diaz, 2017).

Thus, it is suggested that the first phase commences with a video-based training session, so that it is more practical, accompanied by a wiki page containing all the instructions required to conduct UAT, with these tools being available prior to the testing. The staff must also be accessible to help Testers with any necessary explanations.

Prior to doing actual UAT, the Product Owner must determine who will conduct the testing. It is recognised that the inclusion of business roles diversifies the team and ensures that those responsible for testing are aware of the right application usage (Al-Hurmuzi et al., 2018; Otaduy & Diaz, 2017; Pillai & Hemamalini, 2022). Due to the nature of the business and the software applications developed and utilized by ANI, this segregation of roles is sensible, as it also corresponds to the nature of Scrum and the predefined method for writing a user story, as a user is involved.

Following this definition, it is necessary to specify which User Stories will be tested. As stated in Section 4.1, user stories must incorporate both functional and non-functional needs (Al-Hurmuzi et al., 2018; Hu et al., 2021; Padmini et al., 2016; Pandit & Tahiliani, 2015). Therefore, it must be defined by the Product Owner if the testers are to conduct UAT on non-functional user stories.

Figure 8 depicts the flowchart representing the UAT Planning process step.

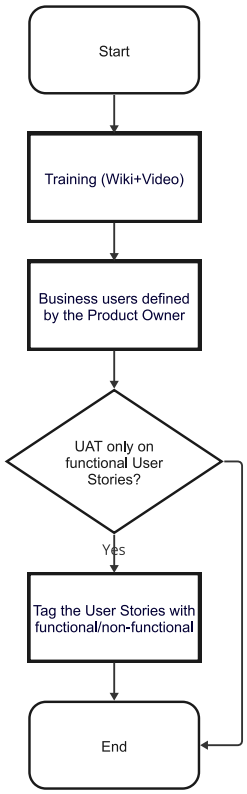

Figure 8 - UAT Planning First Proposal

The **UAT Execution**, carried out by the designated testers, began with the challenge of identifying the primary source of data extraction for testing. As the user stories on the board contained a variety of information, including the user story, acceptance criteria, tags such as bug, functional, and non-functional, and comments, the most efficient way to collect all the data was to export the KanBan board directly using an embedded tool in the Microsoft Planner application, enabling an .xlsx export. This procedure must be performed whenever there is a modification to the planner, so that it remains up to date. This Excel file would also serve as the canvas for registering UAT data and findings. This agrees with the idea provided by Otaduy & Diaz (2017) that information must be updated when new developments occur. Additionally, each sprint's user stories must be distinct from one another, as this facilitates development and UAT (Elallaoui et al., 2015).

Now, the user stories and acceptance criteria in the board's *In Acceptance* bucket must be made ready for testing. This can be accomplished through the use of test diagrams (Elallaoui et al., 2015; Löffler et al., 2010), a conversion from user stories and acceptance criteria to user scenarios and test cases (Aamir & Khan, 2017; Al-Hurmuzi et al., 2018; Pandit & Tahiliani, 2015), or a hybrid of the two, employing mind maps for traceability purposes between user stories, acceptance criteria, and test cases (Otaduy & Diaz, 2017; Rajasekaran & Nithyarao, 2017).

As the personnel at ANI revealed to be resistant to change, the second approach was chosen, which utilized a direct conversion model to convert user stories into user scenarios and acceptance criteria into test cases.

The strategy from Pandit & Tahiliani (2015) was selected due to its simplicity and efficacy, after consideration of the many options presented by the performed Literature Review. For the Tester to have a thorough understanding of what is being tested, the User Stories are utilized as Scenarios and structured as follows (Schwaber & Sutherland, 2020):

[As a] role, [I want] feature, [so that] benefit.

In a similar fashion, acceptance criteria must adhere to a set template so that the meaning of the Acceptance Criteria is clear to the tester.

[Given] input, [When] action, [Then] consequence.

The test cases must then be grouped so that the prioritization is established, and no information is lost, enabling testers to understand how UAT should proceed. This may be accomplished with the notion of Testing Backlog, a Backlog similar to the one used in the Scrum Framework, for test cases (Aamir & Khan, 2017; Geras, 2008).

Those test cases in the Test Backlog must be assigned to a specific business user, who will serve as the designated Tester for those test cases. In the execution, the Tester must do UAT using real data, if available, so that the application is subjected to the real-world inputs that it would have encountered in its intended use (Löffler et al., 2010; Tonkin et al., 2020).

The outcome of the test case must be published in the.xlsx file where the test cases are written, for documentation reasons, describing the tests that were run, as well as the result - *Pass* if the test case performed as expected, or *Fail* if the test produced unexpected results. Several templates and tools empowering testers to organize and perform UAT were described in the literature, and taken into account for the template created (Geras, 2008; Harridon et al., 2021; Hongying & Cheng, 2011; Otaduy & Diaz, 2017; Pillai & Hemamalini, 2022).

If testing is rated as *Pass*, the ticket corresponding to the user story is moved to the *Done* bucket on the Kanban board; if testing is classed as *Fail*, the ticket is transferred to the *Blocked* bucket.

Figure 9 depicts the flowchart representing the UAT Execution process step.

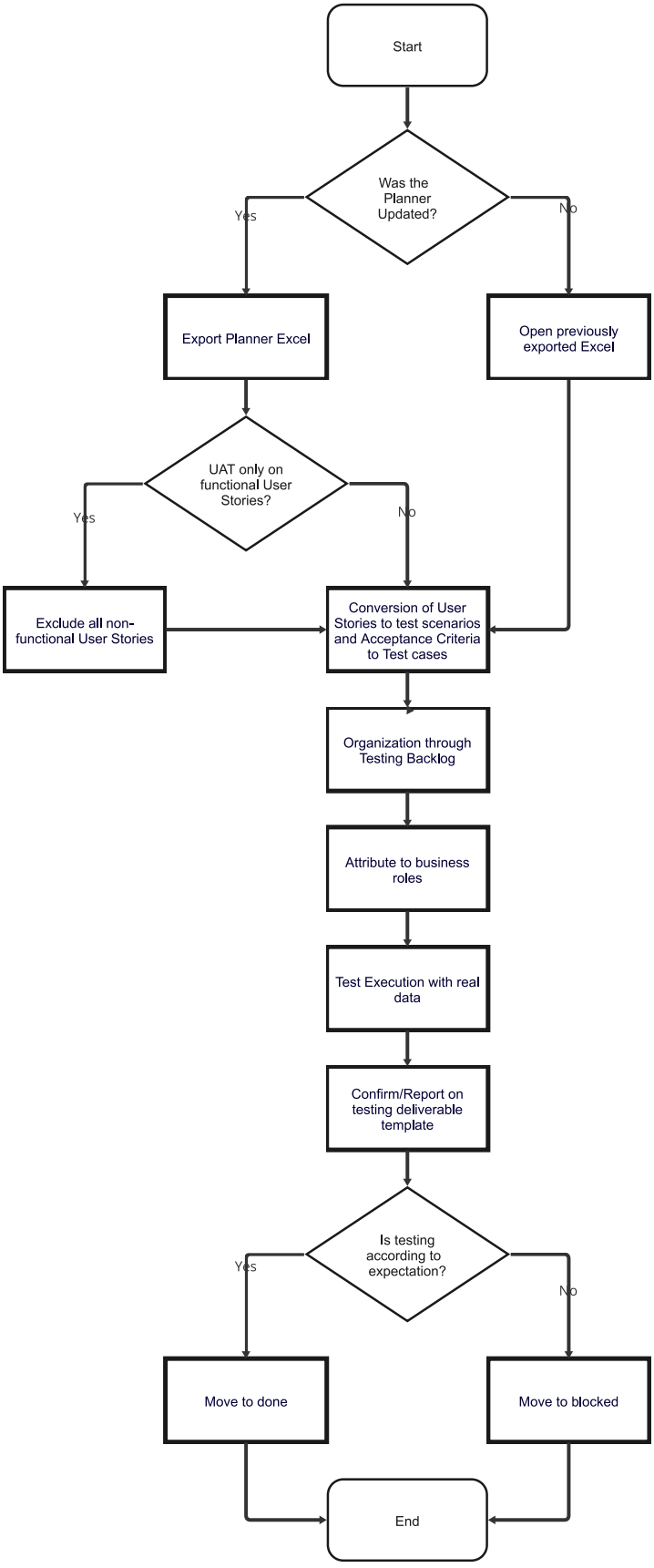

Figure 9 - UAT Execution First Proposal

6.2 UAT Application: DIPI Project

The first iteration process illustrated in Section 6.1 only takes the literature review conducted in Chapter 4 into account. Therefore, the model is premature and needs further development. For the model to be as realistic as possible, testing must be conducted in this phase to eradicate any wrong preconceptions.

The application corresponding to the "DIPI project" was consequently subjected to UAT, one of the least complex application available in the portfolio of Agência Nacional de Inovacão.

Due to the lack of familiarity with the first iteration of the model, as well as UAT itself, testing was conducted following the main defined steps in the model depicted in Figure 8 and Figure 9.

Due to the issues and suggestions discovered through experience and informal meetings with the Product Owner, enhancements were made to facilitate the testing of user stories and increase the process's efficiency.

6.2.1 Testing Sessions

6.2.1.1 First UAT Session

Experience:

After the initial iteration of the process, it is essential to validate and eliminate any biases. For the model to be as accurate to reality as possible, UAT must be undertaken to eliminate any incorrect preconceived ideas.

Due to the lack of familiarity with the first iteration of the model, as well as UAT itself, testing was performed as straightforward as possible, strictly following the defined process steps.

A meeting was held with the Product Owner prior to the UAT to describe the roles/actors that will be utilizing the application and based on the situation, attribute the testing to the actor or actors who may be expecting a certain scenario, as shown in Figure 10:

Actors
Admin
ANI Personnel
NCP Technicians
Delegates

Figure 10 - Actor definition

As the Kanban board is used to organize user stories, non-functional user stories should be able to be filtered off the board if they are not to be tested, as represented in Figure 11.

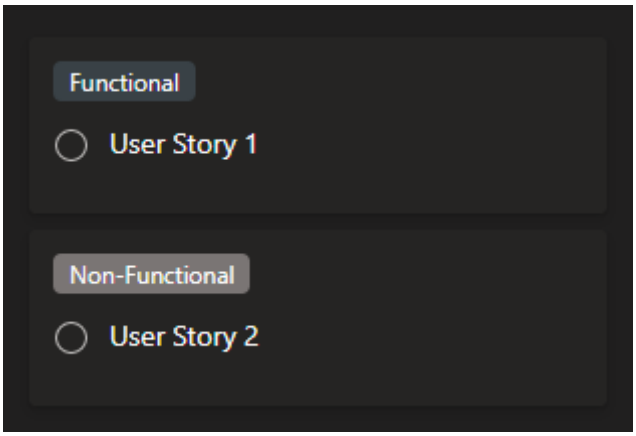

Figure 11 - Functional/Non- Functional User Stories

The user stories written by the product owner were only respective to functional elements of the application, thus there was no need to categorize the user stories on their functional or non-functional attributes.

If non-functional user stories were not to be tested, they had to be removed from the source file so that only the user stories designated for testing were available. The exported source file after data treatment is shown in Figure 12.

Nr	Task Name	Bucket Name	Tags	Completed Checklist Item
2	Upload Resultados - EIC	Done	Functional	
1	Footer	Done	Functional	2/2
3	Notas do projeto	Done	Functional	3/3

AC1	AC2
Criar Footer Block	Usar Footer em todos os ecrans
Criar Tabela	Mostrar notas nos vários ecrans

Figure 12 - Source File after treatment

The testing was then mapped in a spreadsheet, and *Pass* or *Fail* results were determined based on whether the test satisfied the requirements.

A *Lacking Permissions* classification was added, in case that the user conducting the testing only has access to a specific part of the application, although that was used as a reminder and is not considered part of the methodology.

Figure 13 represents an example of the testing process described above, including the test scenarios and test cases:

Nr	Column	Test Scenario / User Story	Test cases / Acceptance Criteria	António
#		[Como] [quero] [para que]	[Dado] [Quando] [Então]	
1		Como admin, quero um footer o para que informação seja mostrada.	Dado qualquer página, quando se clica no footer então este está bloqueado.	
			Dado qualquer página, quando se clica em qualquer página então o footer aparece.	
2		Como admin quero página de upload de resultados funcional para que nova	Dada a página de upload, quando se clica upload from excel file, então página de detalhes de upload é mostrada	
			Dada a página de detalhes de upload, quando se clica em programme part, então lista das temáticas é mostrada.	
			Dada a página de detalhes de upload, quando se clica em year, então lista com anos é mostrada.	
			Dada a página de detalhes de upload, quando se clica em select file, então janela é aberta.	
			Dada a página de detalhes de upload, quando se clica em upload, então upload é iniciado.	
			Dada a página de confirmação de upload, quando a informação é mostrada, então as percentagens PT estão corretas.	
			Dada a página de confirmação de upload, quando a call é identificada, então é igual à inserida.	
3	Projetos	Como NCP Technician quero notas de projeto para que informação seja	Dada página de projeto, quando se carrega num determinado projeto, tabela é mostrada	
			Dados vários ecrãs, quando se carrega num determinado projeto, tabela é mostrada	

Figure 13 - First Testing process Execution

Conclusion:

This process turned out to be not as straightforward as expected. The conversion of user stories and acceptance criteria into test scenarios and test cases was laborious and time-consuming, and the link between them was convoluted and lacked a traceability trail. The lack of familiarity with UAT also proved to be a driver of delay, despite accelerating the more the UAT was progressing. In addition, although the definition of the application's users brought additional value, it was verified a big diversity of permissions inside the application, thus a few tests could not be completed.

Although the exporting of the KanBan board to a spreadsheet seemed like a simpler way to associate the acceptance criteria to the user stories in question, it did not contribute to the agility of testing, as it required unnecessary steps to get the testing set-up.

The usage of a user story standardized form [As a] Role [I want] Feature [So that] Benefit, to depict a scenario revealed helpful to define a scenario, however the standardized form of Acceptance Criteria [Given] Precondition [When] Action [Then] Output, was considered insufficient to be used as Test Cases.

The feedback collected from the Product Owner confirmed the concerns reported above, considering the direct interpretation of the board to be more straightforward and testing itself to be more structured. In addition, although testing was carefully undertaken, the results did not reflect the application's projected day-to-day usage.

6.2.1.2 Second UAT Session

Experience:

The proposed changes of the project for the second session were reached not only from the gathered experience performing UAT, but also from the collected feedback from the product owner in charge of the Project in which the UAT was conducted.

A crucial step was added that is the definition of the most common use cases in which the application will be used. That is a simple, one-time process, that is responsibility of the product owner. That is followed by the association of the roles to the use cases, that will perform the testing assigned to their business role, as shown in Figure 14:

Use Case	Actor	Actions
1	NCP/Delegate	Clicar em Calls na barra lateral
		Escolher um tema específico
		Analisar o concurso/resultados do concurso
		Editar Call
2	NCP/Delegate	Clicar em All Projects na barra lateral
		Escolher Project Data/Proposal Data
		Detalhes da pesquisa (programa, ano, call id, tópico, país, status)
		Verificar os Projetos PT
		Alterar o status do projeto (passa de main list a reserve list, passa de reserve list a main list)

Figure 14 - Roles assigned to Use Cases

Thus, a use case is defined by the Product Owner, containing crucial general steps for the application that must be tested. Then, a number and one of the predefined actors are defined.

There were also changes suggested to the UAT Execution phase of the process. Although the file exportation of the planner seemed like the best way to have all the information concerned to each user story, it did not reveal to be agile to download the planner every time there were updates to the Planner. Thus, the user story information will be gathered directly from the Planner to the testing excel spreadsheet.

As previously identified, the lack of traceability did not allow for a more correct track record of the user stories that were being considered in each scenario in order to write the test cases. That was achieved through a traceability matrix, that may be consulted at any time to make sure the testing is covering every stage of development, and should be completed as soon as a use case suits a user story, as represented in Figure 15:

		Use cases						
		1	2	3	4	5	6	7
User Story Nr	1							
	2		x	x				
	3						x	
	4							x
	5							
	6		x	x	x	x		
	7							x
	8		x					
	9			x				
	10						x	
	11	x						
	12				x	x		

Figure 15 - Traceability Matrix

The columns represent the number attributed to the use cases that were defined by the Product Owner, and the rows represent the User Story numbers. When a User Story is attributed to a use case, an “x” is written.

Regarding the test cases, there was a change concerning the mode on which the test cases are written, as well as the used format. In the first iteration the test cases were being written directly with the acceptance criteria. However, that revealed to not correctly cover the full application. The format used for writing the test cases also suffered a slight change from the first session. The suggested format in the previous one was “[Given] screen X [When] action [Then] output”, causing major confusion to the product owner, as the “Given” did not reveal clarity what was intended. Therefore, the used format in the 2nd iteration was:

[In screen] X [When] action [Then] output

Figure 16 depicts the spreadsheet template used for that purpose:

ID1	ID2	P-N	TC Description	Details	Result	Comments
TC0139	SC7		Na página inicial, quando clico em Upload Results na barra lateral, então a página com 3 opções de upload é mostrada.		Pass	
TC0140	SC7		Na página de Upload Results, quando clico em upload from excel file, então a página de detalhes de upload é mostrada.		Pass	
TC0141	SC7	Neg	Na página de detalhes de upload, quando não coloco qualquer programme part, então mensagem de erro é mostrada.		Pass	
TC0142	SC7	Neg	Na página de detalhes de upload, quando não coloco qualquer year, então mensagem de erro é mostrada.		Pass	
TC0143	SC7	Neg	Na página de detalhes de upload, quando não coloco qualquer call id, então mensagem de erro é mostrada.		Pass	
TC0144	SC7	Neg	Na página de detalhes de upload, quando não coloco qualquer ficheiro de upload, então mensagem de erro é mostrada.		Pass	

Figure 16 - Test Cases Template

The test cases are written, and an ID is attributed. That ID is unique and can only identify one test case. A second ID represents the use case that the test case represents, and there are columns in case the details of the test case ran are relevant, the result of the test case – *Pass* and *Fail* – and for comments on the result in case the test case did not pass.

Additionally, the concept of negative test cases was introduced, as it was identified many acceptance criteria in which negative test cases were deemed necessary. It is also represented in Figure 16 in the third column.

Conclusion:

The alterations made for the second UAT session revealed beneficial. The abandonment of the direct conversion of acceptance criteria to test cases revealed positive, as it was possible to test the application in bigger detail. The use cases better described the actual use of the application, and that complementarity with the acceptance criteria made sure that the whole application could be tested without missing functionalities.

The usage of the *Pass* and *Fail* marks to classify the test case results were deemed insufficient, as there were test cases in which the test did not fail, but a remark should be made to the development team. That is the case, for instance, when there is a formatting error or a lack of an asterisk on a mandatory input field, that even if it does not affect the functionality, must be solved.

Upon meeting with the Product Owner, it was considered that the changes undergone in the second session of UAT facilitated the testing both for users and the product owner. The change in format used for the test cases made clearer what it was going to be tested, the provided deliverable was of easy understanding and what needed corrections was of straightforward interpretation.

Although strongly considering that the methodology presented an improved degree of maturity, some remarks were also made.

Regarding traceability, considered that the implementation was straightforward, however there were user stories that did not fit the parameters or steps defined in the use cases, and in that case more clarity was needed on what to do with those user stories.

Regarding the definition of negativity in the test cases, although the product owner revealed knowledge of the meaning, they did not know in which instances to write them for each test case.

Therefore, those concerns should be addressed in the third session of the UAT process for ANI.

6.2.1.3 Third UAT Session

Experience:

The second session revealed considerably less concerns than the first session, although those should be addressed accordingly. The proposed model for the third and final session will include some missing details that must be corrected.

As previously addressed, the changes on the model were taken from the gathered experience performing UAT in a selected application, as well as from the Product Owner.

As the usage of a binary classification of *Pass* and *Fail* was deemed as insufficient, the *Check* classification was introduced. It provides the Product Owner or the developer with the information that a specific test case did not necessarily failed testing, but attention should be paid to something, that should be covered in the assigned comments. That addition is shown in Figure 17:

ID1	ID2	P-N	TC Description	Details	Result	Comments
TC0152	US4		Na página após o primeiro login, quando o login é feito, então os termos de utilização são mostrados.		Pass	
TC0153	US4	Neg	Na página com os termos de utilização, quando tento entrar na aplicação, então não é possível sem os aceitar.		Pass	
TC0154	US4		Na página com os termos de utilização, quando estes são aceites, então a página inicial é mostrada.		Pass	
TC0155	US7		Na página inicial, quando clico em help na barra lateral, então a página de help é mostrada.		Pass	
TC0156	US7		Na página de help, quando clico nas várias tabs, então as tabs com texto são mostradas.		Pass	
TC0157	US22		Na página inicial, quando clico em home (menu principal), então o footer é mostrado.		Pass	
TC0158	US22		Na página inicial, quando clico em home, então o footer é mostrado.		Check	Não estou c

Figure 17 - Test Cases Template with "Check" addition

The traceability matrix assigning step was also revised, including additional steps that allow users to correctly assign any sort of user story. It was defined that in case a user story did not fit any of the predefined use cases, the possibility the creation of an additional use case should be considered. If it is not the case, a hypothetical use case could be defined for a certain role/actor. That revealed to be the case for the user stories associated with the Administrator role, as it did not fit any of the predefined use cases. Ultimately, in case that is also not possible, the user story remains *Not Assigned*.

Finally, addressing the misunderstanding that arose with the definition of negative test cases, instructions were added, depending on the type of input that the user gives the application.

In the final meeting with the Product Owner from the DIPI Project, it was considered that the process guidelines were detailed, concise and easy to follow for Testers, thus no further suggestions were made to enhance the process.

7. Second Proposal

In this chapter, a Second UAT Process is proposed in Section 7.1, followed by its UAT application in ANI's SGP Project in Section 7.2 and interviews with Product Owners in Section 7.3.

7.1 Experience-Based Second Proposal

The transformation of user stories and acceptance criteria into test scenarios and test cases was ineffective, as it did not cover every step of the application. Therefore, a different strategy was adopted. The Product Owner specifies the most important use cases for the application in a straightforward manner during the Planning phase, as the following example.

In the initial page, click on the login button in the top right corner.

Fill the spaces in blank and click login.

The user profile page opens.

Then, during the planning phase, the use cases are directly assigned to the business users whom the Product Owner deems most qualified to perform them according to their subject matter expertise, as opposed to the first process proposal.

The above example illustrates a use case that encompasses login and profile page tests, which will originate the test cases. In addition to creating test cases, it is necessary to provide each test case a unique ID to facilitate defect reports. The test cases must encompass every page, button, upload, input field, and form of output, such as a generated download. For writing test cases, a modification was made, which helped the to comprehend more effectively what is being tested.

[In the] screen x, [When] action, [Then] consequence.

Now that user stories are not the primary source of test cases, they must be tracked alongside use cases so that the Tester knows what is being tested, as a user story may be included in more than one use case, and a use case may contain more than one user story.

Traceability is a common characteristic of UAT Processes, which can be accomplished through the use of diagrams representing the testing process, tracking test cases to acceptance criteria and user stories (Elallaoui et al., 2015; Pandit & Tahiliani, 2015), or through the use of a traceability matrix, tracking user requirements and test cases (Al-Hurmuzi et al., 2018; Rajasekaran & Nithyarao, 2017).

For tracing the use cases to their relevant user stories, it was adopted the last method. This technique enables the Tester to determine which user stories may be marked as *Done* once UAT for a use case

has been completed. In addition, after the creation of test cases is complete, the Tester must ensure that the test cases include the acceptance criteria for that particular use case, therefore evaluating every relevant component of the product. In the event that a user story is not tracked, a new use case may be established to suit it. In situations where a use case cannot be established, a use case must be developed for a specific role, or a *not-assigned* column must be added to the traceability matrix to accommodate the unassigned user stories, which must still be tested.

In addition, the concept of negative test cases was introduced, enabling the Tester to check that when an invalid input is provided, an error message is presented and unanticipated outputs do not occur (Pandit & Tahiliani, 2015). This is accompanied with the tag *Neg* in the deliverable to ensure that the type of the test case is not overlooked.

After developing test cases based on the defined use cases, the Tester should review the traced user stories and acceptance criteria to ensure that no testing detail is neglected. If so, further test cases must be created.

As the *Pass* and *Fail* classification were deemed insufficient, a new *Check* categorization was introduced to inform the development team that the test case may not have truly failed, but that it may be worthwhile to check something, serving as a warning. That is the case for difficulties such as undetected typos, unformatted parts, screen resolution issues, and others.

Figure 18 and Figure 19 depict the outcomes of the second iteration of the UAT for the IS Department of the ANI. The red boxes display the alterations for the Second Proposal.

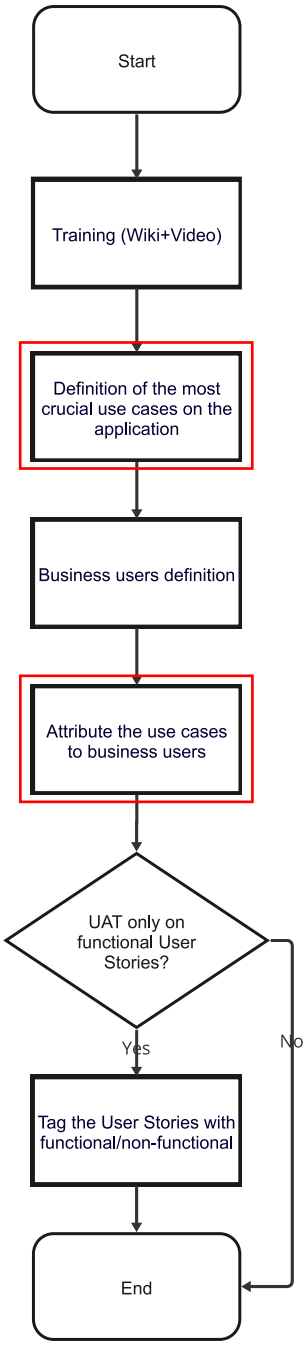

Figure 18 - UAT Planning Second Proposal

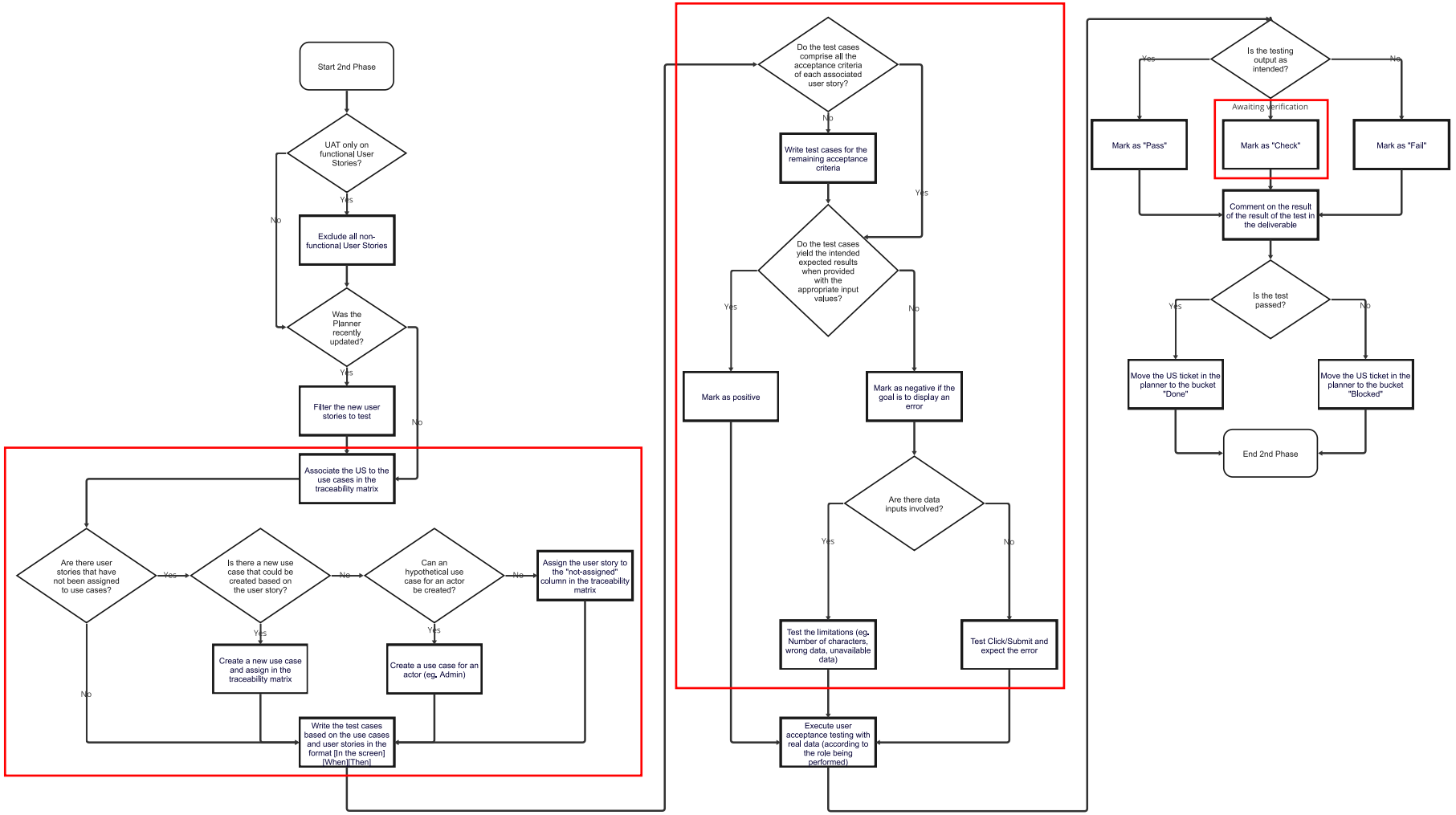

Figure 19 - UAT Execution Second Proposal

7.2 UAT Application: SGP Project

Although the Product Owner of the SGP is an expert on their field, they do not have the same capabilities as the Product Owner from the DIPI Project, and does not perform software development. Therefore, in this project the software development tasks are outsourced to a software development supplier, as referred in Chapter 2.

The UAT process started with the definition of use cases and business roles in the application as the first phase of the proposed process suggests, and acceptance testing was conducted.

UAT process went as smoothly as possible, proving that the process was suited for usage in a more complex application than the one that helped to shape the process.

A small informal meeting was held with the Product Owner, where no remarks were made about the process itself due to their lack of knowledge on the subject. However, it was considered that the way the user acceptance test cases, use cases, results, comments, and template were designed was very comprehensible, thus it could possibly be applied to the project given that appropriate training is provided.

7.3 Feedback Interviews with Product Owners

As mentioned in Chapter 5, feedback interviews were conducted with Product Owners from the various projects that are part of ANI's portfolio. The set of interviews, although intending to evaluate the entire process, it was also aimed at identifying potential changes that might contribute to further enhance the process.

7.3.1 Process Presentation and Feedback Interviews Preparation

In order to validate the model with final users, a series of process presentations and feedback interviews were conducted. Initially, a broad Product Owner meeting with allocated question time was intended to be held. However, it was later determined that not all Product Owners would provide clear insights. Therefore, five participants were gathered, and the meetings were prepared.

First, it was made clear to the participants what the meeting involved and what was intended from them. A presentation on the model was conducted by using the process flowchart, along with examples from the prior execution of the process in both project applications, stopping at every step of the way to make sure that not only the concepts were understood, but also that the steps of the process made sense regarding the UAT in their respective projects. The participants were also asked to interrupt the

presentation in case some concept was not made clear. In the end of the presentation the following questions were the asked:

- Do you consider that the process is applicable to the project you are assigned to?
- Do you consider that you can conduct this process with the provided flowchart?
- Is there anything you would like to add?

Finally, it was defined that the data should be analysed by creating categories that demonstrate the most important facts about the methodology:

- General Principle
- Negative Testing
- Execution Time
- Traceability
- Tester Training
- Planning Testing
- Test Case Result
- Concept Familiarity

From the interpretation of the annotations collected in every meeting, these categories were then classified on a scale from 1 to 5, on the following Likert Scale:

1. Not Valuable
2. Slightly Valuable
3. Moderately Valuable
4. Very Valuable
5. Extremely Valuable

The results will be presented using a table and radar chart in the subsequent section.

7.3.2 Process Presentation and Feedback Interviews Results

The presentation of the project provided insights about how the Product Owners – users of the process – value certain categories that are present in the methodology. That was done thorough an interpretation of the questions about the subjects, as well as the comments made about certain steps of the process.

Remarks were made to induce the interviewees to talk about the whole process, touching on the categories defined in the section above.

Various remarks were made along with the session that helped to validate the proposed steps for UAT for ANI's software applications. The concept of negative test cases was unknown to the interviewees, although when explained it was concluded that it was performed, even if not extensively. Interviewee

nr. 1 stated that negative testing is often forgotten, and that is why including it in the process along with how and when to run it would be of great importance, due to the nature of ANI's applications.

The traceability matrix was mentioned to be useful by there out of five participants, helping to keep track of which user stories were completely tested.

The most concerning remark provided by the product owners was that the process seemed very time consuming, which combined with the tasks that the personnel is already assigned to would be a demanding task. However, interviewee Nr. 5 presented the opposite opinion that even if it looks very time-consuming, ultimately time will be saved due to the structure of UAT, as unnecessary testing will not be con-ducted. By being more effective, bugs will be detected sooner, and the quality of the software deployed will be much higher.

Moreover, interviewee nr 4 indicated a lack of definition on how the test cases are communicated to the product team, suggesting that the procedure when test cases fail needs to be better defined. That motivated changes depicted in the third iteration in Chapter 8.

At the end of the presentation, as described in the previous section, the three defined questions were asked. All the product Owners considered that the UAT process was applicable to their project, improving overall testing process and the quality of software deployed. However, interviewee Nr. 5 expressed that in his opinion, due to the nature of the project, the definition of some of the use cases would be more time consuming, considering that the usage of scenarios would contribute to the simplicity of the testing process.

In a similar fashion, all of them thought that the Flowchart provided was simple to follow and a great way to conduct the UAT process. However, it was stated that by the majority that even if the IS department personnel are familiar with Agile and Agile practices and Scrum, the business users are not. For that reason, training sessions would be required so that UAT is performed efficiently as defined by the proposed process. Product Owners Nr. 2 and 4 noted that training would be required for them-selves as well, as they lack UAT knowledge.

Finally, all the participants considered that the existing of a template tailored to ANI's UAT needs will greatly facilitate the overall process. The following table and average radar graph containing the results from that perception are displayed in Table 8 and Figure 20.

Table 8 - Feedback Results

Interviewee	Nr. 1	Nr. 2	Nr. 3	Nr. 4	Nr. 5
General Principle	5	5	4	4	5
Negative Testing	5	2	4	3	4
Execution Time	2	4	3	3	5
Traceability	4	5	2	5	5
Tester Training	4	5	2	5	5
Planning Testing	5	2	4	3	3
Test Case Result	5	5	5	4	4
Concept Familiarity	4	5	3	4	4

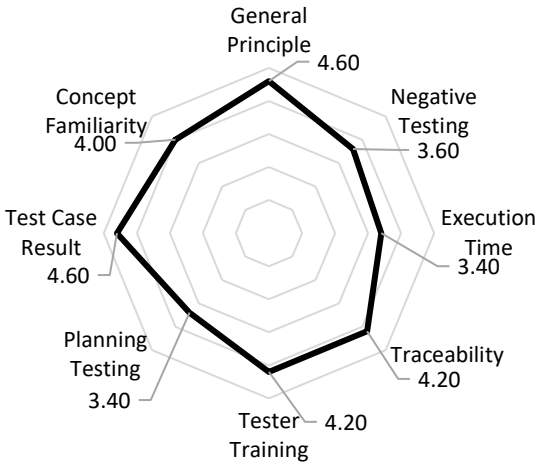

Figure 20 - Average Feedback Results

As it can be seen from the analysis of the radar chart, testers consider that the execution time and the ability to plan UAT to be the less valuable characteristics from the UAT process. However, in most of the categories it scores higher than 4 – between very valuable and extremely valuable – thus considering that the various aspects of the model are appropriate.

It was captured from one of the interviews that the process lacked definition on how the test cases that were classified as *Check* or *Fail* would be communicated to the development team. After a reflexion, that concern was deemed pertinent and needed to be addressed.

Therefore, it was determined that in case the user story has failed use cases, the test case IDs should also be included in the user story ticket to indicate to the development team exactly what should be fixed, along with being put in the *Blocked* bucket, as shown is Figure 21.

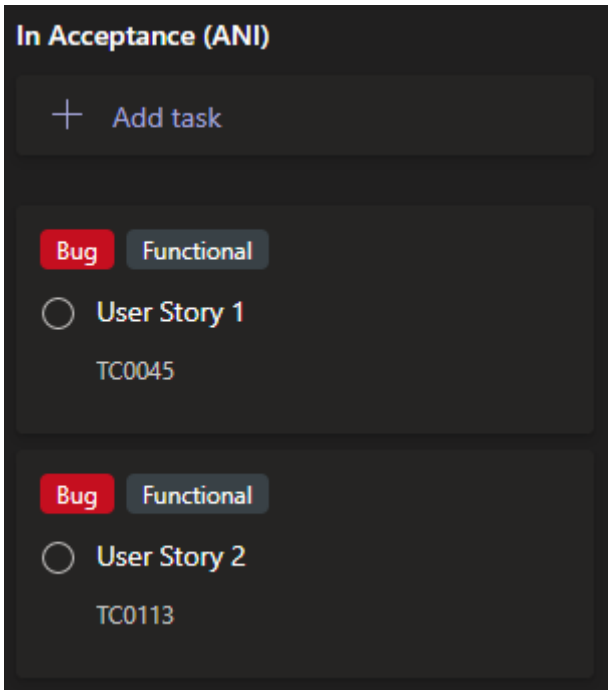

Figure 21 - User Story tags

Also, in the UAT template that was provided, it was included an automatic filtered spreadsheet containing the user stories that did not pass the testing process, so that the identification is made efficiently, as represented in Figure 22:

ID1	ID2	P-N	TC Description	Details	Result	Comments (Tester)
TC0002	SC1		Na página de login e registo como perito, quando clico em registo com autenticaçã		Check	Botão de registo com autentica
TC0014	SC1	Neg	Na página de registo de perito, quando não carrego na checkbox de açoes de mon		Check	Permite o registo, no entanto n
TC0027	SC1	Neg	Na página de registo de perito, quando seleciono um ficheiro para anexar que não		Fail	Deixa anexar qualquer formato
TC0038	SC2	Neg	Na página de criação de fichas de avaliação, quando não coloco título e clico em cr		Check	Falta asterisco de obrigatorieda
TC0088	US17		Na página de filtragem de peritos, quando clico nas opções avançadas de filtragen		Fail	Não se encontra nas opções de
TC0089	US17		Na página de filtragem de peritos, quando clico nas opções avançadas de filtragen		Fail	Não se encontra nas opções de
TC0099	SC3	Neg	Na página de filtragem de um perito, quando seleciono um perito, então o email		Fail	O email é sempre mostrado
TC0100	SC3	Neg	Na página de filtragem de um perito, quando seleciono um perito, então o contac		Fail	O contacto telefónico é sempre

Figure 22 - Development Team Filtered Spreadsheet

8. Final Proposal and Discussion of Results

This Chapter will present the final proposal of the UAT process for ANI in Section 8.1, as well as a discussion of the quantitative results from the UAT Applications in the DIPI and SGP Projects in Section.

8.1 Final Proposal

The second iteration process depicted in Chapter 7 arose from UAT experience and the Product Owner's contributions to the DIPI Project. The method is far more mature than the initial iteration, however it was observed that testing on a new, more complicated project, namely the SGP Project or Expert Management System, was necessary to demonstrate the process's viability.

Based on the knowledge gained from the UAT sessions that were done on the SGP project application, one process enhancement was determined. When the test case is classified as *Pass*, the process becomes tedious and time-consuming due to the need to comment on the test case. As a result, the necessity to remark on passed test cases was eliminated, as it did not contribute much to the overall process.

After the UAT process was completed, interviews with Product Owners of the various projects in the ANI's portfolio revealed a lack of effectiveness on how a failed test case is communicated to the development team.

As a result, a step was introduced that required the test case ID to be included in the user story that is placed in the *Blocked* bucket, as well as an enhancement to the given deliverable template, that allowed the development team to directly view the *Fail* or *Check* test cases.

The third and final iteration of the proposal is depicted in Figure 23 and Figure 24. The alterations for the final proposal are contoured with a red box. The UAT Planning phase did not suffer any alterations from the Second Proposal.

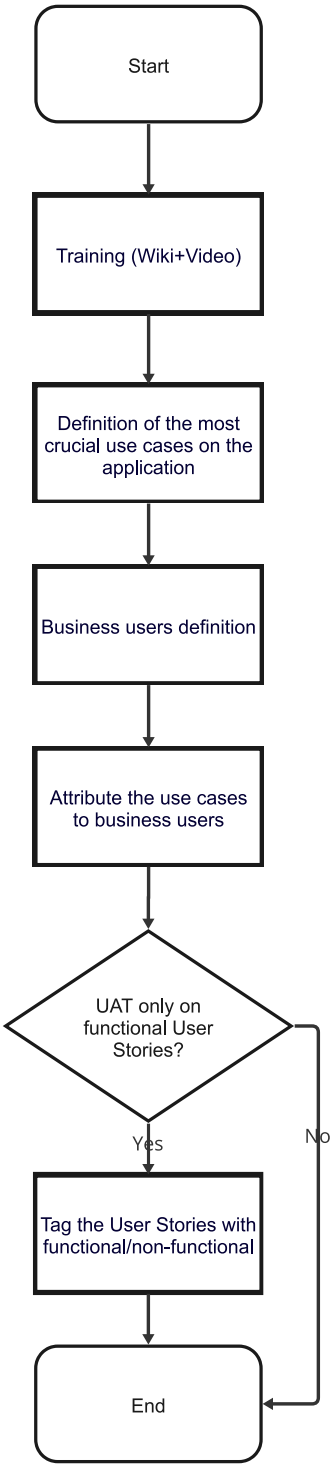

Figure 23 - UAT Planning Final Proposal

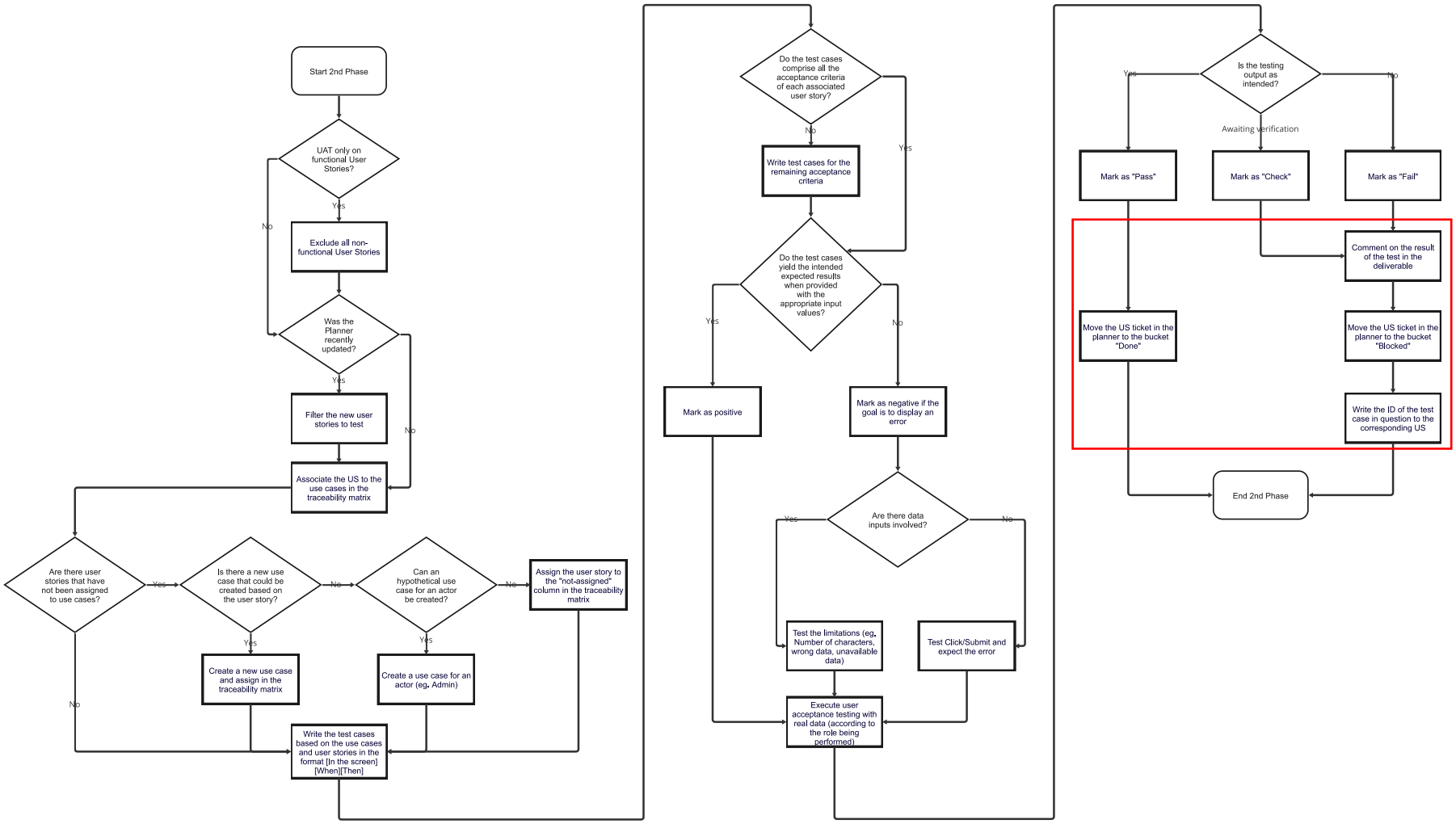

Figure 24 - UAT Execution Final Proposal

8.2 Discussion of Results

After discussing the qualitative data gathered from experience and the personnel at ANI, it is crucial to discuss the results from the UAT both on the DIPI and SGP project's software applications.

Given that there is a lack of data from previous UAT, an approximation of the number of user stories that were sent back to the software development supplier for corrections was estimated. The analysis, also shown in Section 2.4.2, is dated from October 2022 and its summary is shown in Table 9.

Table 9 - Software Development Issues

Project	Done US	US w/ Bugs	% US w/ Bugs
SIFIDE	45	7	16%
SGI	38	15	39%
SGP	**19**	**9**	**47%**
Orçamentos	13	-	-
Deslocações	2	-	-
Total	**117**	**31**	**26%**

It can be concluded by the analysis of the table, the quality of software deployed by the software development supplier is poor. An average of 26% of User Stories are deployed with bugs, being blocked and sent back to the development team for correction. The average value is deflated, as there is no data regarding user stories containing bugs in projects C and D.

Table 10 - UAT Test Cases Data

Project	DIPI	SGP
Total Test Cases	174	218
Pass	164	204
Check	1	6
Fail	9	8
% Issues	**5.75%**	**6.86%**

As previously stated, UAT was conducted both in the DIPI and SGP projects, and the results regarding test cases are stated in Table 10. The percentage of issues – test cases that either failed or need to be checked – is remarkably similar.

In the DIPI Project, the user stories that were assessed corresponded to a part of the application that was already developed and tested to its full extent; therefore, all the user stories were in the bucket *Done*.

Regarding the SGP Project, the use cases that were selected by the Product Owner comprised of user stories in the *Done* and *In Acceptance* buckets.

Table 11 details the results from the overall UAT process, by project.

Table 11 - UAT Application Results

	Project	
	DIPI	**SGP**
Nr. User Stories	23	22
% Software Developed	100.0%	35.5%
Nr. *Fail* Tests	9	8
Nr. *Check* Tests	1	6
US w/ *Fail* Test Cases	4	5
US w/ *Check* Test Cases	1	2
% *Blocked* User Stories	**21.7%**	**31.8%**

´

As it can be concluded from Table 11, there were clear problems found in the software developed. Although the percentage of test cases with issues was not very high – around 6% for both projects – that becomes more concerning when it is considered that the vast majority of user stories tested had already undergone UAT.

When those use cases are associated with user stories, the problems become clearer. Considering both *Fail* and *Check* user stories, the percentage of *Blocked* user stories in the DIPI and SGP projects was greater than 20 and 30 percent, respectively.

Table 9 clarifies the problems that were had at the time with the quality of software deployed, with 47% of SGP user stories with bugs. Although issues were found in the UAT performed by the team, still a huge percentage of issues are found in the software. The DIPI project has no data in that regard, however, the percentage of *Blocked* user stories is closer to the average percentage of user stories with bugs in the firm, although it must be consider that UAT had already been performed.

For that reason, it can be stated that the proposed UAT process for ANI clearly contributes to the efficacy of UAT, by finding a large number of bugs in the software that were not identified during the previous sessions conducted in the company.

9. Conclusion

9.1 Contributions

The primary goal of this research was to develop a UAT process for ANI, while on an internship as Project Management Assistant. It involved multiple steps in order to reach a feasible solution to improve UAT. The company context gathered through experience, contributed to understand the environment to be studied, as well as finding opportunities for improvement where the impact of the proposal would be greater. By studying the environment, it was clear that the main concerns were not only on the software quality, but also on the experience of the personnel with Agile, Scrum and UAT approaches that could be addressed through this proposal.

Various methodologies were understood through a background study on Software Development Methodologies, so that the process could be better tailored to the current usage, or new perspectives could be found. The literature review on UAT aided to learn the leading approaches on UAT in the Agile methodology, providing tools that could be used in the UAT Proposal. It was found that either the literature provided UAT ideas for extremely specific instances or the proposals were overly general.

Hence, a UAT process proposal for ANI was developed, through three iterations. Those proposals were assisted by the company's personnel, and UAT was performed in two different projects, while taking into account not only the environment of the company, but also the existing tools. The process was enhanced in every iteration by attempting various practices, so that it would work as intended – an efficient and effective UAT process for ANI's project applications.

Qualitative data was collected from Product Owners, which helped to validate the process, considering that the practices and values from process would aid the company to ultimately achieve higher software quality. Quantitative data from the experience gathered while performing UAT also supported the process validation, as it was found that more that 20% of user stories on both projects – the majority classified as *Done* – still contained defects or some sort of warning issue.

This new proposal brought improvements to the whole UAT process in the company, as it filtered the software defects to a higher degree then the approach that was previously in place. It contributed for the enhancement of efficiency and effectiveness of UAT in ANI through a tailored process that suits their needs and helps to perform UAT on a more structured manner. Similar organizations or companies that provide software with the same characteristics could also benefit from the usage of this proposal. Ultimately, it contributes to the existing body of knowledge on UAT by providing a different perspective on the topic, through an iterative thought process that prioritizes the company.

9.2 Limitations

Some limitations of the proposed process should be addressed. The most critical limitation of the proposed process is the fact that the process is tailored to ANI and based on its context, portfolio, characteristics, and personnel, and therefore it may be not as applicable to other companies or projects. However, it may be directly applicable to companies in the same field. Additionally, it was limited to the authorized by the firm, hence compromises had to be made to in that regard, not being possible to implement by introducing a different software.

The process was based on two different projects of the ANI's portfolio, so that biases could be removed and as extensively applicable as possible. However, there can be some projects that the process can be not as applicable as the ones that were tested. Moreover, there was only one Tester performing the process due to time and resource constraints.

Finally, the process also relied greatly on interviews with the company personnel that are not experts on UAT, and the qualitative data and results from those interviews are subjected to interpretation.

9.3 Future Research

Further research could be developed on the implementation of the UAT long term benefits. As the conducted research was time-constrained, it was not possible to evaluate the long-term effects on the quality of the software developed, as well as the benefits of this research on the company's environment.

Investigation could also be done on the potential benefits of implementing automation methods on the process, as it focuses on manual UAT. That could contribute to reducing the time on repetitive tasks requiring manual effort, therefore becoming a more efficient pro-cess.

Furthermore, a study could be conducted on the applicability of the proposed UAT process not only for other companies, but also on the feasibility of adaptations to other software development methodologies.

Finally, a study could be conducted on the possibility of adaptation of the UAT process to usability testing, integrating user interaction characteristics.

References

Aamir, M., & Khan, M. N. A. (2017). Incorporating quality control activities in scrum in relation to the concept of test backlog. *Sādhanā*, *42*(7), 1051–1061. https://doi.org/10.1007/s12046-017-0688-7

Al-Hurmuzi, S., Al-Khanjari, Z., & Al-Kindi, I. (2018). Proposed Feasible PEF framework for User Acceptance Testing. *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, 242–248. https://doi.org/10.1109/CSIT.2018.8486225

Almeida, F., Simões, J., & Lopes, S. (2022). Exploring the Benefits of Combining DevOps and Agile. *Future Internet*, *14*(2), 63. https://doi.org/10.3390/fi14020063

Archi – Open Source ArchiMate Modelling. (n.d.). Retrieved 28 June 2023, from https://www.archimatetool.com/

Bass, L., Weber, I. M., & Zhu, L. (2015). *DevOps: A software architect's perspective*. Addison-Wesley Professional.

Bassil, Y. (2012). A Simulation Model for the Waterfall Software Development Life Cycle. *International Journal of Engineering*, *2*(5), 7.

Beck, K., Beedle, M., van Bennekum, A., Cunningham, W., Fowler, M., Greening, J., Highsmith, J., Hunt, A., Jeffries, R., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*. http://agilemanifesto.org/

Bertram, D. (2006). *Likert Scales*. *CPSC 681 – Topic Report*, 1–10.

Best, B. de. (2017, July 8). *DevOps Plan—The deliverables*. IT Strategy. https://en.itpedia.nl/2017/07/08/devops-plan-the-deliverables/

Bhavsar, K., Shah, Dr. V., Research Guide, Computer Science & Engineering, Indus University, Ahmedabad, India., Gopalan, Dr. S., & Research Co-Guide, Business Administration & Management, Indus University, Ahmedabad, India. (2020). Scrum: An Agile Process Reengineering In Software Engineering. *International Journal of Innovative Technology and Exploring Engineering*, *9*(3), 840–848. https://doi.org/10.35940/ijitee.C8545.019320

Cagan, M. (2017). *INSPIRED: How to create tech products customers love* (Second edition). Wiley.

Chopra, A., Prashar, A., & Sain, C. (2013). Natural Language Processing. *International Journal of Technology Enhancements and Emerging Engineering Research (IJTEEE)*, *1*(4), 131–134.

de França, B. B. N., Jeronimo, H., & Travassos, G. H. (2016). Characterizing DevOps by Hearing Multiple Voices. *Proceedings of the XXX Brazilian Symposium on Software Engineering*, 53–62. https://doi.org/10.1145/2973839.2973845

Dima, A. M., & Maassen, M. A. (2018). From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management. *Journal of International Studies*, *11*(2), 315–326. https://doi.org/10.14254/2071-8330.2018/11-2/21

Ebert, C. (2007). The impacts of software product management. *Journal of Systems and Software*, *80*(6), 850–861. https://doi.org/10.1016/j.jss.2006.09.017

Ebert, C. (2014). Software Product Management. *IEEE Software*, *31*(3), 21–24. https://doi.org/10.1109/MS.2014.72

Ebert, C., & Brinkkemper, S. (2014). Software product management – An industry evaluation. *Journal of Systems and Software*, *95*, 10–18. https://doi.org/10.1016/j.jss.2013.12.042

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*, *33*(3), 94–100. https://doi.org/10.1109/MS.2016.68

Elallaoui, M., Nafil, K., & Touahni, R. (2015). Automatic generation of UML sequence diagrams from user stories in Scrum process. *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, 1–6. https://doi.org/10.1109/SITA.2015.7358415

ERP Public Sector SNC-AP - Software de Contabilidade Pública. (2023). PRIMAVERA BSS. https://pt.primaverabss.com/pt/software/solucoes-setoriais/administracao-publica/public-sector/

Fahmy, S., Haslinda, N., Roslina, W., & Fariha, Z. (2012). Evaluating the Quality of Software in e-Book Using the ISO 9126 Model. *International Journal of Control and Automation*, *5*(2), 115–122.

Felizardo, K. R., Mendes, E., Kalinowski, M., Souza, É. F., & Vijaykumar, N. L. (2016). Using Forward Snowballing to update Systematic Reviews in Software Engineering. *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–6. https://doi.org/10.1145/2961111.2962630

Freeman, E. (2019). *Devops for dummies* (1st ed). John Wiley and Sons.

Geras, A. (2008). Leading Manual Test Efforts with Agile Methods. *Agile 2008 Conference*, 245–251. https://doi.org/10.1109/Agile.2008.73

Gonçalves, L. (2018). Scrum. *Controlling & Management Review*, *62*, 40–12.

Harridon, M., Harun, M. K., Ahmad, A. A., Mohd Hashim, Z. H., Mohd Aris, K. D., Abdul Latif, B. R., Wasi Zainal Ariffin, M., Yaakop, A., Jalal Amran, M., & I, M. (2021). User Acceptance Test of Static Flight Simulator Boeing 737-800 of Universiti Kuala Lumpur MIAT. *International Journal of Scientific and Research Publications (IJSRP)*, *11*(11), 293–297. https://doi.org/10.29322/IJSRP.11.11.2021.p11936

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *34*(9), 120–127. https://doi.org/10.1109/2.947100

Hongying, G., & Cheng, Y. (2011). *A customizable agile software Quality Assurance model*. 382–387.

Hooda, I., & Singh Chhillar, R. (2015). Software Test Process, Testing Types and Techniques. *International Journal of Computer Applications*, *111*(13), 10–14. https://doi.org/10.5120/19597-1433

Hron, M., & Obwegeser, N. (2018). *Scrum in Practice: An Overview of Scrum Adaptations*. Hawaii International Conference on System Sciences. https://doi.org/10.24251/HICSS.2018.679

Hu, P., Chaowen, C., Ma, Y., & Wang, X. (2021). Acceptance Testing Optimization Method for Continuous Delivery. *2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT)*, 168–173. https://doi.org/10.1109/CECIT53797.2021.00037

Introduction: ArchiMate® 3.2 Specification. (n.d.). Retrieved 21 April 2023, from https://pubs.opengroup.org/architecture/archimate3-doc/

Kittlaus, H.-B., & Fricker, S. A. (2017). *Software Product Management: The ISPMA-Compliant Study Guide and Handbook* (1st ed. 2017). Springer Berlin Heidelberg : Imprint: Springer. https://doi.org/10.1007/978-3-642-55140-6

Kramer, M. (2018). Best Practices in Systems Development Lifecycle: An analyses based on the Waterfall Model. *Review of Business & Finance Studies*, *9*(1), 77–84.

Larman, C., & Basili, V. R. (2003). Iterative and Incremental Development: A Brief History. *IEEE Computer Society*.

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). *Software Development Life Cycle AGILE vs Traditional Approaches*. *37*.

Löffler, R., Güldali, B., & Geisen, S. (2010). Towards Model-based Acceptance Testing for Scrum. *Softwaretechnik-Trends*. https://www.semanticscholar.org/paper/Towards-Model-based-Acceptance-Testing-for-Scrum-L%C3%B6ffler-G%C3%BCldali/71684fd5efaa91ca1b0d40a5f8bb12adbde82f6d#citing-papers

Long, R. (2020). *Human Centric User Acceptance Testing*. Pacific Northwest Software Quality Conference (PNSQC).

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015). Dimensions of DevOps. In C. Lassenius, T. Dingsøyr, & M. Paasivaara (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (Vol. 212, pp. 212–217). Springer International Publishing. https://doi.org/10.1007/978-3-319-18612-2_19

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016). Relationship of DevOps to Agile, Lean and Continuous Deployment. In P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, & T. Mikkonen (Eds.), *Product-Focused Software Process Improvement* (Vol. 10027, pp. 399–415). Springer International Publishing. https://doi.org/10.1007/978-3-319-49094-6_27

McGreal, D., & Jocham, R. (2018). *The professional product owner: Leveraging Scrum as a competitive advantage*. Addison-Wesley.

Microsoft Planner. (2023). https://tasks.office.com/

Mohammad, S. M. (2017). DevOps Automation and Agile Methodology. *International Journal of Creative Research Thoughts*, *5*(3), 946–949.

Munassar, N. M. A., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues*, *7*(5).

Mushtaq, Z., & Qureshi, M. R. J. (2012). Novel Hybrid Model: Integrating Scrum and XP. *International Journal of Information Technology and Computer Science*, *4*(6), 39–44. https://doi.org/10.5815/ijitcs.2012.06.06

.NET | Build. Test. Deploy. (2023). Microsoft. https://dotnet.microsoft.com/en-us/

Otaduy, I., & Diaz, O. (2017). User acceptance testing for Agile-developed web-based applications: Empowering customers through wikis and mind maps. *Journal of Systems and Software*, *133*, 212–229. https://doi.org/10.1016/j.jss.2017.01.002

Padmini, K. V. J., Perera, I., & Dilum Bandara, H. M. N. (2016). Applying agile practices to avoid chaos in User Acceptance Testing: A case study. *2016 Moratuwa Engineering Research Conference (MERCon)*, 96–101. https://doi.org/10.1109/MERCon.2016.7480122

Pandit, P., & Tahiliani, S. (2015). AgileUAT: A Framework for User Acceptance Testing based on User Stories and Acceptance Criteria. *International Journal of Computer Applications*, *120*(10), 16–21. https://doi.org/10.5120/21262-3533

Parizi, R. M., Lee, S. P., & Dabbagh, M. (2014). Achievements and Challenges in State-of-the-Art Software Traceability Between Test and Code Artifacts. *IEEE Transactions on Reliability*, *63*(4), 913–926. https://doi.org/10.1109/TR.2014.2338254

Petersen, K., Wohlin, C., & Baca, D. (2009). The Waterfall Model in Large-Scale Development. In F. Bomarius, M. Oivo, P. Jaring, & P. Abrahamsson (Eds.), *Product-Focused Software Process Improvement* (Vol. 32, pp. 386–400). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02152-7_29

Petricioli, L., & Fertalj, K. (2022). Agile Software Development Methods and Hybridization Possibilities Beyond Scrumban. *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, 1093–1098. https://doi.org/10.23919/MIPRO55190.2022.9803402

Pillai, N. S. R., & Hemamalini, R. R. (2022). Hybrid User Acceptance Test Procedure to Improve the Software Quality. *The International Arab Journal of Information Technology*, *19*(6). https://doi.org/10.34028/iajit/19/6/14

Pillai, N. S. R., Hemamalini, R. R., Padmavathy, V., & S., N. (2019). Framework for Multiple User Acceptance Testing to Avoid Chaos. *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, 1–6. https://doi.org/10.1109/ICSCAN.2019.8878803

Power Automate | Microsoft Power Platform. (n.d.). Retrieved 28 June 2023, from https://powerautomate.microsoft.com/en-us/

Product Management Framework | Pragmatic Institute. (n.d.). Retrieved 19 April 2023, from https://www.pragmaticinstitute.com/framework/

Rajasekaran, K., & Nithyarao, T. K. (2017). The Effective New Frame Work for Mind Mapping Matrix Test Case Techniques. *International Journal of Innovative Science and Research Technology*, *2*(7), 470–474.

Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses* (1st ed). Crown Business.

Roche, J. (2013). Adopting DevOps practices in quality assurance. *Communications of the ACM*, *56*(11), 38–43. https://doi.org/10.1145/2524713.2524721

Rong, G., Jin, Z., Zhang, H., Zhang, Y., Ye, W., & Shao, D. (2019). DevDocOps: Towards Automated Documentation for DevOps. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 243–252. https://doi.org/10.1109/ICSE-SEIP.2019.00034

Royce, W. W. (1970). Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON*, *26*, 328–388.

Saeed, S., Jhanjhi, N. Z., Naqvi, M., & Humayun, M. (2019). Analysis of Software Development Methodologies. *International Journal of Computing and Digital Systems*, *8*(5), 445–460. https://doi.org/10.12785/ijcds/080502

Sanjaya, H., & Andry, J. F. (2021). Quality assurance of project management information system with ISO 9126. *Bulletin of Social Informatics Theory and Application*, *5*(2), 82–87.

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum*. Prentice Hall.

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. 1–14.

Shah, J., Dubaria, D., & Widhalm, J. (2018). A Survey of DevOps tools for Networking. *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 185–188. https://doi.org/10.1109/UEMCON.2018.8796814

Smeds, J., Nybom, K., & Porres, I. (2015). DevOps: A Definition and Perceived Adoption Impediments. In C. Lassenius, T. Dingsøyr, & M. Paasivaara (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (Vol. 212, pp. 166–177). Springer International Publishing. https://doi.org/10.1007/978-3-319-18612-2_14

Software de Apresentação de Diapositivos do Microsoft PowerPoint | Microsoft 365. (n.d.). Retrieved 28 June 2023, from https://www.microsoft.com/pt-pt/microsoft-365/powerpoint

Software de Folha de Cálculo do Microsoft Excel | Microsoft 365. (n.d.). Retrieved 28 June 2023, from https://www.microsoft.com/pt-pt/microsoft-365/excel

Stray, V., Moe, N. B., & Aasheim, A. (2019). Dependency Management in Large-Scale Agile: A Case Study of DevOps Teams. *Hawaii International Conference on System Sciences.*

The Visual Collaboration Platform for Every Team | Miro. (n.d.). Https://Miro.Com/. Retrieved 28 June 2023, from https://miro.com/

Tonkin, E. L., Nieto, M. P., Bi, H., & Vafeas, A. (2020). Towards a Methodology for Acceptance Testing and Validation of Monitoring Bodyworn Devices. *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 1–6. https://doi.org/10.1109/PerComWorkshops48775.2020.9156257

Trimble, J., & Webster, C. (2013). From Traditional, to Lean, to Agile Development: Finding the Optimal Software Engineering Cycle. *2013 46th Hawaii International Conference on System Sciences*, 4826–4833. https://doi.org/10.1109/HICSS.2013.237

Trivedi, P., & Sharma, A. (2013). A Comparative Study between Iterative Waterfall and Incremental Software Development Life Cycle Model for Optimizing the Resources Using Computer Simulation. *2013 2nd International Conference on Information Management in the Knowledge Economy*.

Tuteja, M., & Dubey, G. (2012). A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models. *International Journal of Soft Computing and Engineering (IJSCE)*, *2*(3).

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a Reference Framework for Software Product Management. *14th IEEE International Requirements Engineering Conference (RE'06)*, 319–322. https://doi.org/10.1109/RE.2006.66

Visualização de Dados | Microsoft Power BI. (2023). https://powerbi.microsoft.com/pt-pt/

What is Low-Code | Low-Code Guide. (2023). https://www.outsystems.com/guide/low-code/

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 1–10. https://doi.org/10.1145/2601248.2601268

Zhu, H., Zhou, M., & Seguin, P. (2006). Supporting Software Development With Roles. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *36*(6), 1110–1123. https://doi.org/10.1109/TSMCA.2006.883170

Appendix

Appendix A – Survey Questions to the Product Owners

Category	Question
As a Product Owner	**Q1** How do you evaluate your performance as PO, from the beginning of the project until now?
	Q2 Did you feel motivated in the performance of your duties as an PO?
	Q3 As an OP, did you feel supported in the project?
	Q4 As PO, how satisfied are you with the time you have allocated to the project?
	Q5 Do you consider that the project is going as planned?
About the Supplier's Team	**Q6** How do you rate your relationship with the supplier's team?
	Q7 How satisfied are you with the collaborative nature of the supplier's team?
	Q8 Was the supplier's team committed to the project?
	Q9 Were you satisfied with the supplier's performance?
	Q10 Did the supplier's deliveries meet the desired quality?
	Q11 Do you consider that the supplier carries out exhaustive tests?
	Q12 Did the supplier's deliveries meet the established deadlines?
	Q13 Was the supplier's team available for clarifications when needed?
About the Project Management Team	**Q14** Do you consider the *Planner* an important tool for the development of the project?
	Q15 Do you consider that the project risk identification and assessment exercise was relevant?
	Q16 Do you consider that you need additional training for the practices currently used?
	Q17 Do you consider that the support of the ANI IS team was useful during the execution of the project?
Others	**Q18** What importance do you attribute to the Demo-Days?