User Acceptance Testing Process for a Digital Public Service

António Fernandes
Instituto Superior Técnico
antoniojorgefernandes@tecnico.ulisboa.pt

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.

I. INTRODUCTION

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 and the Scrum Framework (Beck et al. 2001, Schwaber and 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, there is a 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. Testing is 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 research 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 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 ensure 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 plays a relevant role in the in the digital transformation and transition it has set out to achieve.

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 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.

Supported by prior contextualization of ANI's current state, data and ANI personnel, developing a tailored UAT process for ANI is the motivation behind this research.

.

II. RESEARCH CONTEXT

A. 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. Utilizing a customized Kanban Board, allows for 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 tickets 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, 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.

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 they maintain constant contact via telephone, online meetings, and email.

B. Opportunities for Improvement

The daily expertise obtained at ANI was essential to grasp ANI's current situation. I was also crucial to understand 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, 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 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.

Findings:

- Product owners are concerned about the time allocated for their project, contributes to various issues, in particular relating to UAT.
- There are significant concerns about the software development supplier, including dissatisfaction with their collaborative approach, performance, and adherence to project timelines.
- The product owners believe that general training in project management techniques is needed for the project management team.
- The Product Owners are not aware of their responsibilities, project management approaches, and the development of user stories and acceptance criteria.
- Product Owners suggest improved project scheduling assistance, shorter development sprints for better supervision, and enhanced user acceptance testing efforts.
- Compliance with project management approaches varies among product owners, causing challenges for the project management team.
- The software development supplier faces challenges in communication, understanding user stories, meeting deadlines, and delivering quality software.

The identified issues in project management methodologies and UAT require further research and exploration for future improvements in the IS department.

III. RESEARCH BACKGROUND

To understand the different Project Management methodologies, it is necessary to comprehend them and how the differences in Roles, Activities and Deliverables affect the Software Development Life Cycle. Therefore, some of the most used Software Development Methodologies were compared: Waterfall, Scrum, DevOps and Product Management.

A. Roles

The roles differ both in number and in their tasks, according to the goals of the methodology. There is a focus on Planning and Quality assurance in the Waterfall methodology, product and team management in the Scrum methodology, pipeline and communication in DevOps and providing customers with the best possible product is the focus of the roles in Product Management.

B. Activities

The activities in the Waterfall methodology are sequential and one cannot start before the other finishes. Scrum and DevOps both focus on continuous integration and deployment. Scrum has a set of activities that must be repeated in a pre-defined time frame of 2-4 weeks to ensure that the framework works as intended, covering all phases of software development life cycle. DevOps consist more of a series of principles that must be followed to shorten the gap between Development and Operations. Project Management activities focus on product discovery and business innovation.

C. Deliverables

Table 1 describes the main differences of deliverables in terms of quantity and type for each approach.

Table 1 - 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

IV. LITERATURE REVIEW

Although there is plenty of investigation available on software testing, there is a lack of research regarding manual UAT in the Agile or Scrum methodology. Using forward and backward snowballing approaches, additional articles were selected based on suitable characteristics for UAT.

Aamir and 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.

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.

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, they suggest 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.

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. 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.

Pillai and 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 after each testing phase. This entails the introduction of a "test case document initiation" for the conversion of requirement documents to test cases, detailing document history and reference documents to be consulted. These 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.It is asserted that by using the hybrid model, the number of errors found has increased, costs decrease, and consequently, software quality improves.

Hongying and 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.

Elallaoui et al. (2015) recommends the creation of automatically generated UML diagrams, by the inclusion of an algorithm in the Scrum framework.

Pandit and 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). Information is extracted from user stories and acceptance criteria using a predefined template, and translated to document object model in XML. 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.

Otaduy and 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 – both 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".

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. Along with that, 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.

The results from the UAT Literature Review are summarized in Table 2.

Table 2 - Approach and Characteristics features of UAT Literature Review

	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	
(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 & Hemamalini, 2022)	X		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)

V. UAT PROCESS

The research methodology used for the UAT Process for ANI is shown in Figure 1.

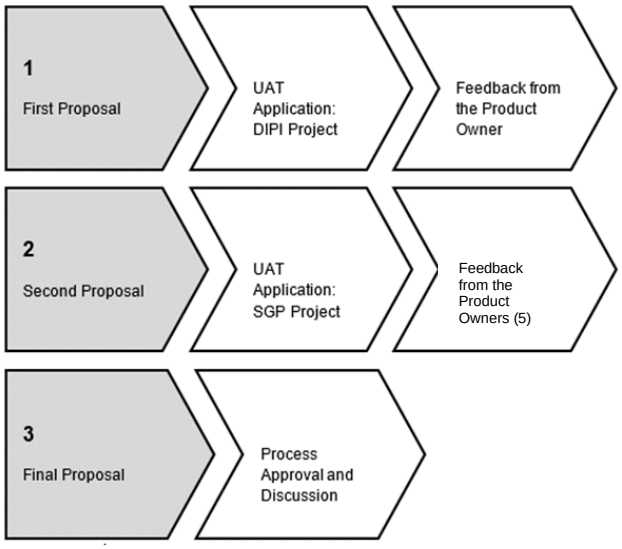

Figure 1 - Research Methodology

A. First Proposal

UAT Planning

Agile and Scrum teams may utilize a variety of strategies for UAT. These processes contain extremely valuable concepts that can be used by any firm or product. 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 by using a KanBan board for user story structure and prioritization, with the development team being an external supplier. Although Scrum is one of the most popular soft ware 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.

Therefore, 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.

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 responsible for inspiring the team to do UAT.
- The addition of a Tester to the Scrum Team. The testers are company 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 phase, UAT Execution, is conducted by the Tester.

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 towards 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 and Cheng 2011). Padmini et al. (2016) agree that feedback sessions were 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 contributed to personnel's improved ability to manage software (Harridon et al. 2021, Otaduy and 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 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 and Diaz 2017, Pillai and 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. User stories must incorporate both functional and non-functional needs (Al-Hurmuzi et al. 2018, Hu et al. 2021, Padmini et al. 2016, Pandit and Tahiliani 2015). Therefore, it must be defined by the Product Owner if the testers are to conduct UAT on non-functional user stories.

UAT Execution

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 and 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 boards 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 and Khan 2017, Al-Hurmuzi et al. 2018, Pandit and Tahiliani 2015), or a hybrid of the two, employing mind maps for traceability purposes between user stories, acceptance criteria, and test cases (Otaduy and Diaz 2017, Rajasekaran and 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 and 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 and 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 and 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 and Cheng 2011, Otaduy and Diaz 2017, Pillai and 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.

B. UAT Application: DIPI Project and Feedback from the Product Owner

Three UAT sessions were performed in the DIPI Project of ANI, each one followed by a meeting with the project's Product Owner.

Those sessions revealed issues within the first iteration of the process. 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 to that, although the definition of the application's users brought additional value, a big diversity of permissions inside the application was verified, thus some 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 to be 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 second UAT session indicated that the usage of Pass and Fail mark in case the testing output was as intended was deemed insufficient, as there were cases in which the test did not fail, although a remark should be made to the development team. That is the case when there is a formatting error, a lack of an asterisk on a mandatory input field, that does not affect the functionality, yet that must be solved. Also, regarding traceability, it was 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 with what to do with those user stories.

With regards to 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.

The third UAT session corrected the aforementioned concerns, being considered by the Product Owner that the process guidelines were detailed and concise, thus easy to follow for Testers, and no further suggestions were made to enhance the process.

C. *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.

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 due to their subject matter expertise, as opposed to the first iteration of the process.

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 for each function of the web pages, it is necessary to provide each test case a unique ID to facilitate defect reports. It must encompass every page, button, upload, input field, and form of output, such as a generated download. For writing test cases, a modified version of the template was used, which helped the Tester comprehend what is being tested more effectively:

[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 usage of diagrams representing the testing process, tracking test cases to acceptance criteria and user stories (Elallaoui et al. 2015, Pandit and Tahiliani 2015), or through the usage of a traceability matrix, tracking user requirements and test cases (Al-Hurmuzi et al. 2018, Rajasekaran and Nithyarao 2017).

For tracing the use cases to their relevant user stories, ANI adopted this final 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 the test cases is completed, 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. If 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 fictional 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 and 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 was deemed insufficient, the result classification was also enhanced. 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 relevant, serving as a warning. This is the case for difficulties such as undetected typos, unformatted parts, screen resolution issues, and others.

D. *UAT Application: SGP Project, Feedback from the Product Owner*

Even though the Product Owner is an expert on their field, they do not have the same capabilities as the Product Owner from the DIPI Project and do not perform software

development, therefore the software development tasks are outsourced to a software development supplier.
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 and comments, and template were designed was very comprehensible.

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 to indicate to the development team exactly what should be fixed, along with being put in the "Blocked" bucket. Also, in the UAT template that was provided, an automatic filtered spreadsheet containing the user stories that did not pass the testing process was included, so that the identification is made efficiently.

E. Final Proposal

The second iteration process arose from the 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, 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, a series of interviews with Product Owners of the various projects in the portfolio of ANI 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 process for the UAT Planning and UAT Execution phases are depicted in Figure 2 and Figure 3 respectively.

VI. PROCESS APPROVAL AND DISCUSSION

A. Feedback Presentation and Interviews

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.
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.
Remarks were made to induce the interviewees to talk about the whole process, touching on predefined categories.

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. An interviewee 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, an interviewee 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 conducted. By being more effective, bugs will be detected sooner, and the quality of the software deployed will be much higher.

Moreover, another interviewee indicated a lack of definition on how the test cases are communicated to the product team, suggesting that what to do when test cases fail needs to be better defined, motivating changes in the final proposed process.

At the end of the presentation, 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, one of the interviewees 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

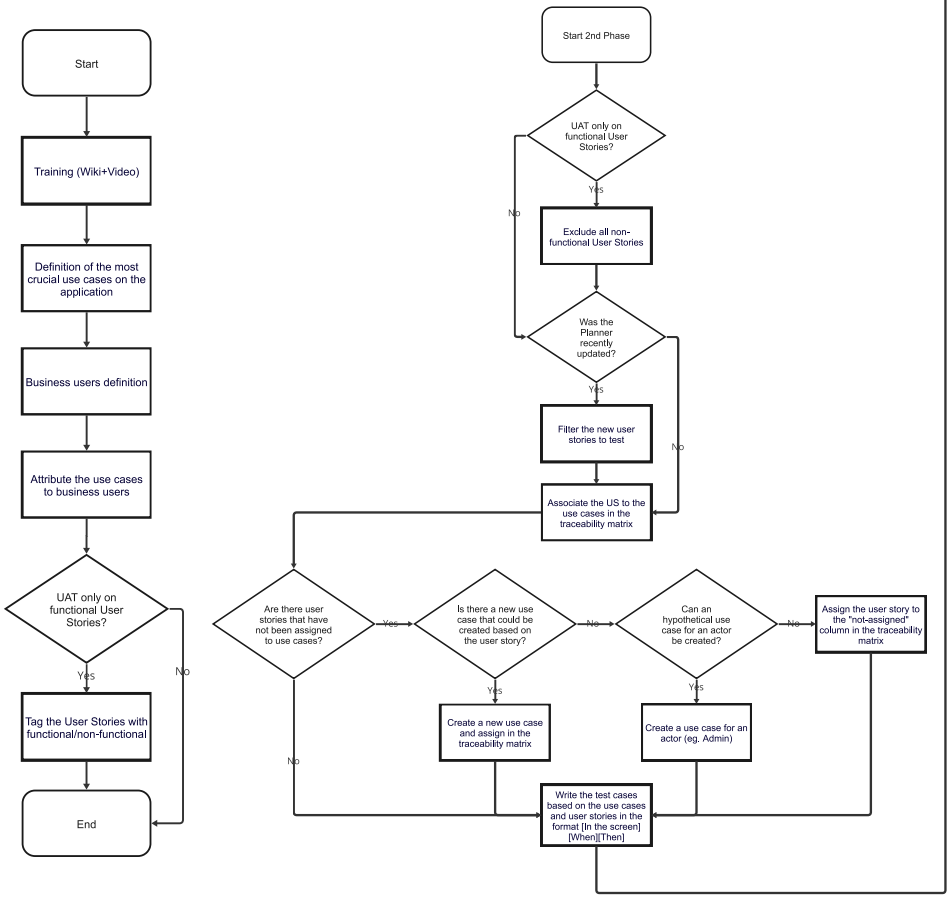

Figure 2 - Final Proposal UAT Planning

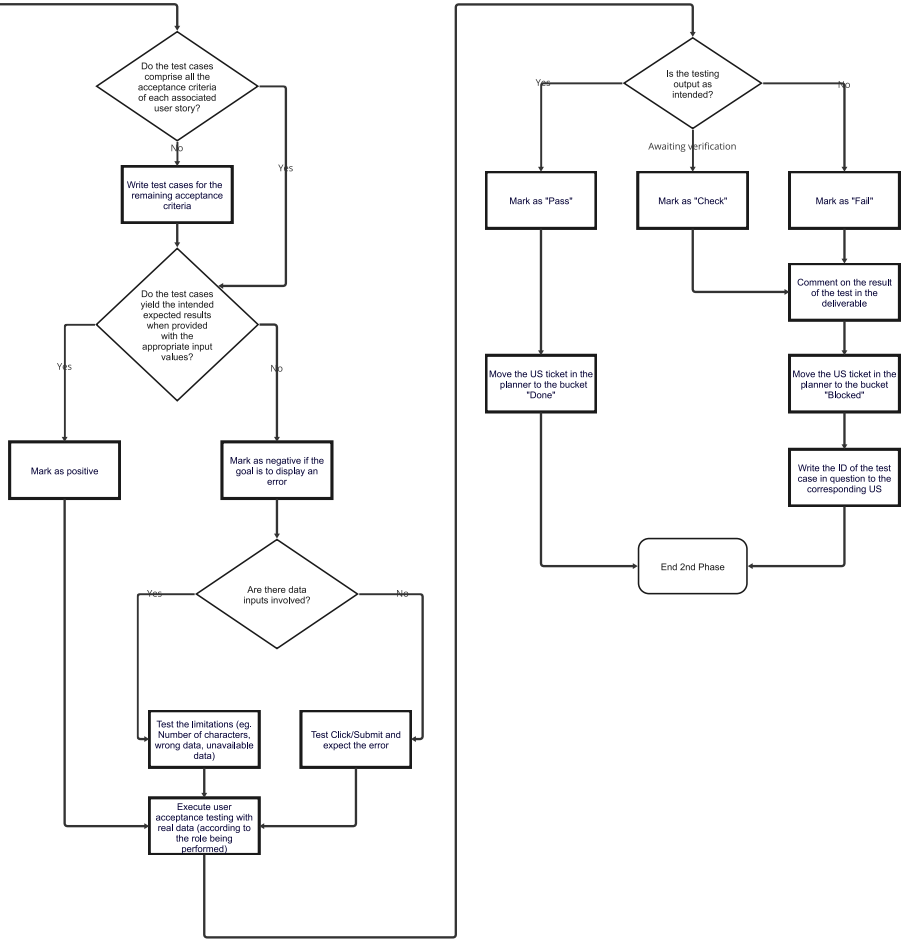

Figure 3 - Final Proposal UAT Execution

defined by the proposed process. Two of the Product Owners noted that training would be required for themselves as well, as they lack UAT knowledge.

Figure 4 depicts the average feedback results from the Product Owners on predefined categories. The results are on a scale from one to five, five being the highest degree of importance attributed by the Product Owner.

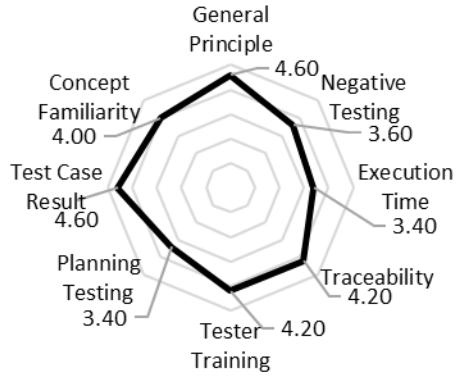

Figure 4 - Average Feedback Results

B. 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 quality of software deployed by the software development supplier is poor, as an average of 26% of User Stories were deployed with bugs, being blocked and sent back to the development team for correction.

UAT was conducted both in the DIPI and SGP projects, and the results regarding test cases are stated in Table 3. The percentage of issues – test cases that either failed or need to be checked – is remarkably similar.

Table 3 - 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%**

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".

With respect to the SGP Project, some crucial use cases were selected by the Product Owner, comprising of user stories in the "Done" and "In acceptance" buckets.

As it can be drawn from Table 3, 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.

Table 4 details the overall results from the UAT process, by project.

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 4 - UAT Experience 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%**

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.

VII. CONCLUSIONS AND FUTURE RESEARCH

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 resources that could be used in the UAT Proposal.

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 projects.

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 – that were mostly 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.

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 process.

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.

VIII. REFERENCES

Aamir M, Khan MNA (2017) Incorporating quality control activities in scrum in relation to the concept of test backlog. *Sādhanā* 42(7):1051–1061.

Al-Hurmuzi S, Al-Khanjari Z, AI-Kindi I (2018) Proposed Feasible PEF framework for User Acceptance Testing. *2018 8th Int. Conf. Comput. Sci. Inf. Technol. CSIT*. (IEEE, Amman), 242–248.

Anon (2023) Microsoft Planner. Retrieved (April 18, 2023), https://tasks.office.com/.

Beck K, Beedle M, van Bennekum A, Cunningham W, Fowler M, Greening J, Highsmith J, et al. (2001) Manifesto for Agile Software Development. Retrieved (April 18, 2023), http://agilemanifesto.org/.

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

Chopra A, Prashar A, Sain C (2013) Natural Language Processing. *Int. J. Technol. Enhanc. Emerg. Eng. Res. IJTEEE* 1(4):131–134.

Elallaoui M, Nafil K, Touahni R (2015) Automatic generation of UML sequence diagrams from user stories in Scrum process. *2015 10th Int. Conf. Intell. Syst. Theor. Appl. SITA*. (IEEE, Rabat), 1–6.

de França BBN, Jeronimo H, Travassos GH (2016) Characterizing DevOps by Hearing Multiple Voices. *Proc. XXX Braz. Symp. Softw. Eng.* (ACM, Maringá Brazil), 53–62.

Geras A (2008) Leading Manual Test Efforts with Agile Methods. *Agile 2008 Conf.* (IEEE, Toronto, ON, Canada), 245–251.

Harridon M, Harun MK, Ahmad AA, Mohd Hashim ZH, Mohd Aris KD, Abdul Latif BR, 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. *Int. J. Sci. Res. Publ. IJSRP* 11(11):293–297.

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

Hu P, Chaowen C, Ma Y, Wang X (2021) Acceptance Testing Optimization Method for Continuous Delivery. *2021 2nd Int. Conf. Electron. Commun. Inf. Technol. CECIT*. (IEEE, Sanya, China), 168–173.

Löffler R, Güldali B, Geisen S (2010) Towards Model-based Acceptance Testing for Scrum. *Softwaretechnik-Trends*.

Otaduy I, Diaz O (2017) User acceptance testing for Agile-developed web-based applications: Empowering customers through wikis and mind maps. *J. Syst. Softw.* 133:212–229.

Padmini KVJ, Perera I, Dilum Bandara HMN (2016) Applying agile practices to avoid chaos in User Acceptance Testing: A case study. *2016 Moratuwa Eng. Res. Conf. MERCon*. (IEEE, Moratuwa, Sri Lanka), 96–101.

Pandit P, Tahiliani S (2015) AgileUAT: A Framework for User Acceptance Testing based on User Stories and Acceptance Criteria. *Int. J. Comput. Appl.* 120(10):16–21.

Pillai NSR, Hemamalini RR (2022) Hybrid User Acceptance Test Procedure to Improve the Software Quality. *Int. Arab J. Inf. Technol.* 19(6).

Rajasekaran K, Nithyarao TK (2017) The Effective New Frame Work for Mind Mapping Matrix Test Case Techniques. *Int. J. Innov. Sci. Res. Technol.* 2(7):470–474.

Royce WW (1970) Managing the Development of Large Software Systems. *Proc. IEEE WESCON* 26:328–388.

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

Tonkin EL, Nieto MP, Bi H, Vafeas A (2020) Towards a Methodology for Acceptance Testing and Validation of Monitoring Bodyworn Devices. *2020 IEEE Int. Conf. Pervasive Comput. Commun. Workshop PerCom Workshop*. (IEEE, Austin, TX, USA), 1–6.