Practical Findings from Applying Quality Assurance Activities in the Development of Three Information Systems for Power Companies

Geraldo Braz Junior[1][a], Luis Rivero[1][b], João Almeida[1][c], Simara Rocha[1][d], Aristófanes Silva[1][e], Anselmo Paiva[1][f], Carlos Castro[1], Darlan Quintanilha[1][g], Italo Santos[1][h], Erika Alves[2] and Samira Barbosa[2]

[1]*Núcleo de Computação Aplicada, Universidade Federal do Maranhão, São Luis, Brazil*

[2]*Equatorial Energia S/A, São Luis, Brazil*

Keywords: Software Quality, Experience Report, Lessons Learned, Power Company Information Systems.

Abstract: Quality can be achieved in software development by identifying and fixing defects, improving the development process and including configuration management activities. However, for novice development teams, including the above activities may be difficult when developing large or complex information systems. Thus, to gain insight on how to improve software quality, novice software engineers may review reports from real software development projects and apply lessons learned. In this paper, we report how software engineering activities for quality assurance were adapted within three power company information system projects. We explain how activities regarding version tracking, software testing and user interface design were carried out by three novice software development teams within a software organization of about 40 collaborators. Our results indicate that version control can be costly at first, but is useful to assess the current state of the development of features. Furthermore, though low-cost evaluation and design approaches, the end product can meet users' needs and reduce rework when launching a version of an information system.

1 INTRODUCTION

Current research indicates the many benefits of applying quality assurance activities such as (Rothenberger et al., 2010): successful and cost-effective software production, better product quality, user satisfaction, cost reduction, rework reduction and meeting goals in time. However, software engineers still indicate several challenges in the implementation of quality assurance activities such as losing focus when dealing with large teams; and difficulty in adapting methods or practices into the development life-cycle (Kamei et al., 2017).

To the best of our knowledge, there are few reports on how quality assurance activities were performed and the lessons learned from such experiences. Furthermore, the available reports are not always related to the development of information systems nor provide details of the applied process (Wonohardjo et al., 2019). Considering the above, in this paper we describe how three quality assurance activities were performed within the context of three different projects developing information systems. In these projects, at least one of the following methods/practices were implemented: (a) exploratory testing; (b) version control strategy with gitflow; and (c) usability evaluation and redesign. During the execution of the projects, we took notes on the process, positive outcomes and improvement opportunities of the activities we carried out in an agile-based context.

2 RELATED WORK

Researchers and practitioners have reported their experiences on how quality assurance procedures have

[a] https://orcid.org/0000-0003-3731-6431
[b] https://orcid.org/0000-0001-6008-6537
[c] https://orcid.org/0000-0001-7013-9700
[d] https://orcid.org/0000-0003-3318-7281
[e] https://orcid.org/0000-0003-0423-2514
[f] https://orcid.org/0000-0003-4921-0626
[g] https://orcid.org/0000-0001-8134-4873
[h] https://orcid.org/0000-0002-2041-7538

Braz Junior, G., Rivero, L., Almeida, J., Rocha, S., Silva, A., Paiva, A., Castro, C., Quintanilha, D., Santos, I., Alves, E. and Barbosa, S.
Practical Findings from Applying Quality Assurance Activities in the Development of Three Information Systems for Power Companies.
DOI: 10.5220/0011081700003179
In *Proceedings of the 24th International Conference on Enterprise Information Systems (ICEIS 2022) - Volume 2*, pages 159-166
ISBN: 978-989-758-569-2; ISSN: 2184-4992

been applied (Oprins et al., 2019). For instance, Hron and Obwegeser (2018) discuss the results from a systematic literature review reporting the challenges and motivations that lead to modifications in an agile software development process. The authors identified some strategies for adaptation such as: (a) introduction of additional processes, artifacts or roles; (b) inclusion of notes or instructions to better guide the development process; and (c) implementation of changes in procedures, artifacts or roles. The results are useful for understanding the motivations for including changes, such as considering usability in software development, or combining further methods to improve the results of the development project.

In the field of software testing, Collins and Lucena (2012) shared their experience of applying software testing automated tools. After describing two software development projects and reporting the procedures for carrying out the test automation using Selenium, the authors present lessons learned regarding: (a) collaboration within the development team; (b) fitting the testing tools according to the test strategy; (c) degree of automation according to project stages; (d) simplification of the automation process; and (e) types of tests that could be automated.

With regards to usability improvement, de Carvalho et al. (2016) analyzed how user centered design is explored within an agile context. To do so, the authors carried out a systematic literature review and presented an experience report on how usability evaluations can be applied in development. The evaluations were performed before the development of the software, yielding less rework, usability improvement and a change on the mindset of the team.

The above reports suggest an interest in describing how adaptations within a software development process can support the quality improvement of information systems under development. Yet, there is still a need for further reports on the use of different quality assurance approaches in further contexts, since some reports do not describe in which settings these approaches were adopted, their strengths and weaknesses from the point of view of software engineering practitioners (Theocharis et al., 2015). For instance, some papers discussing how adaptations are being made, do not describe the contexts in which they occur neither the specifics of the changes implementation (Hron and Obwegeser, 2018). Furthermore, to the best of our knowledge, although there are works describing the inclusion of specific activities in practice (Hassani-Alaoui et al., 2020) (Ruane et al., 2020), they mostly provide an overview of the changes; while others that focus on specific changes related to quality assurance could provide further examples of the artifacts that were specifically developed by the software organization (Collins and de Lucena, 2012) (de Carvalho et al., 2016). Thus, in this paper we present further examples of how software engineering technologies and procedures have been included for the quality assurance of information systems. Below, we present details of our agile-intended software development projects and what changes were made.

3 THE SOFTWARE DEVELOPMENT PROJECTS

Three software development projects were carried out for the development of information systems to help solve problems in the context of power companies. The Equatorial National Power Company hired the Computing Group at Federal University of Maranhão for this task. The Computing Group had around 40 collaborators and worked in the development of information systems using artificial intelligence and embedding machine learning (ML) systems. Below, we present each project chronologically (from the oldest to the most recent), providing details on their purpose, how they were executed and the implemented changes for quality assurance.

3.1 Project A - Tracking Clients with High Degree of Lawsuit Chance

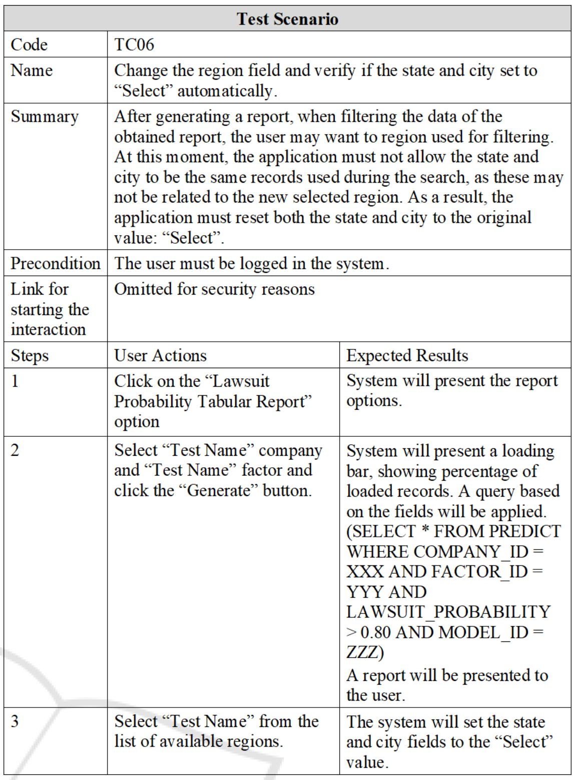
Context: The goal of this project was to develop an information system for automatically identifying the predisposition of consumers in filing lawsuits towards power companies. Based on attributes such as power outage, improper power cut, improper billing and increased power consumption, a score indicating the probability of a client to file a lawsuit was embedded into a database, providing details of the dissatisfaction of clients. With this information, the information system could generate reports so that the customer service of the company could provide support to these clients and improve their satisfaction.

Project Details: This project lasted 18 months (from 2018 to 2020). In all, the development team had 24 software engineers. The project followed an agile development process, following practices from the Scrum methodology (Kamei et al., 2017), but focusing on deliverables for each milestone within the project. The following goals were set within the project: (a) Scope and architecture definition (Months 1-3); (b) Development of Data Base (Months 4-10); (c) Development of Module I, which would analyze the data to export managerial reports (Months 4-10);

(d) Development of Module II, which would optimize algorithms for automatically generating and adapting clients groups (Months 6-14); (e) Development of User Interface that would integrate the data and produce reports (Months 12-18); and (f) Testing, Integration and Release (Months 6-18). Each sprint lasted for a month and the goals between sprints overlap to improve productivity. Also, although there were 24 software engineers, they were split into smaller teams to meet development goals concurrently, such as algorithm, architecture and interface modeling, implementation and testing of features, implantation of the system and others.

Adaptations for Quality Assurance: Since the project had several updates regarding the user interface requirements and handling of the data to provide reports, there was no updated documentation for the testing of the web application. As a result, the development team decided to employ a manually exploratory testing approach, in which the tester actively controls the design of the tests as those tests are performed; and uses information gained while testing to design new and better tests (Quesada-López et al., 2019). We planned the tests by identifying and organizing the different functionalities of the web application through meetings with the development team and scrum master, who had validated the requirements with the product owner.

The testing team derived a test scenario with an initial path to test how each function would work in an expected way. Once such path was was covered, alternative paths considering different inputs were considered in order to identify further paths. Figure 1 shows an example of a test scenario that was derived through the exploratory testing. The identification of scenarios and development of test scenarios continued during the development process in the final sprints. At the end of each sprint, new test scenarios would be discussed and the old ones would be reviewed, updated (if necessary) and re-executed to verify the proper functioning of the ML web application.



Test Scenario		
Code	TC06	
Name	Change the region field and verify if the state and city set to "Select" automatically.	
Summary	After generating a report, when filtering the data of the obtained report, the user may want to region used for filtering. At this moment, the application must not allow the state and city to be the same records used during the search, as these may not be related to the new selected region. As a result, the application must reset both the state and city to the original value: "Select".	
Precondition	The user must be logged in the system.	
Link for starting the interaction	Omitted for security reasons	
Steps	User Actions	Expected Results
1	Click on the "Lawsuit Probability Tabular Report" option	System will present the report options.
2	Select "Test Name" company and "Test Name" factor and click the "Generate" button.	System will present a loading bar, showing percentage of loaded records. A query based on the fields will be applied. (SELECT * FROM PREDICT WHERE COMPANY_ID = XXX AND FACTOR_ID = YYY AND LAWSUIT_PROBABILITY > 0.80 AND MODEL_ID = ZZZ) A report will be presented to the user.
3	Select "Test Name" from the list of available regions.	The system will set the state and city fields to the "Select" value.

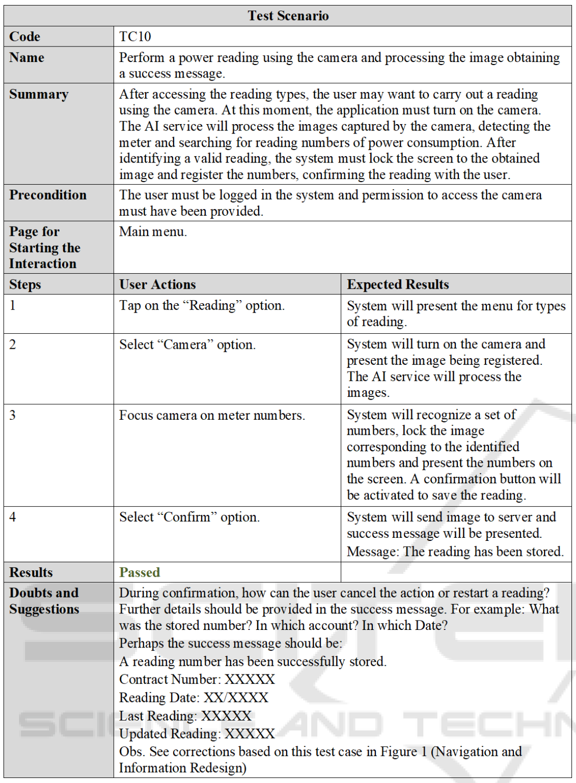
Figure 1: Test scenarios for a functionality from the ML web application.

3.2 Project B - Self Power Consumption Reading

Context: The goal of this project was to develop a mobile information system that could register power consumption readings. By using the application, clients would be able to register their consumption in three different ways: by image, by voice and by keyboard. The power company wanted users to have different ways of measuring their consumption, as this solution was originally designed for rural areas that are difficult to access.

Project Details: This project lasted for 24 months (from 2019 to 2021). The development team had 14 team members and followed a similar development process as in project A. Along with the functionality tests, we applied usability ad-hoc inspections and validations, to identify improvement opportunities in the design of the mobile application.

Adaptations for Quality Assurance: In this project, there was a need to identify usability problems and propose redesign suggestions to meet them. To achieve this goal, we followed a usability evaluation approach similar to the one by Plechawska et al. (2013). A usability and UX consultant was contacted to retest the entire system and also carry out an ad-hoc usability inspection. The consultant reexecuted the identified test scenarios and during each test, he verified usability attributes. Since the consultant had more than 5 years of experience evaluating the usability of web and mobile applications, he did not need to check usability heuristics as performed by Plechawska et al. (2013), thus characterizing the inspection as ad-hoc (Damian et al., 2020). In this sense, during the evaluation process, if usability principals from mobile applications were not met during the interaction of the analyzed test scenario from the point of view of the consultant, a suggestion/doubt was included in the test report and it was discussed with the development team and the representatives from the power company. Figure 2 shows an example of the tests scenarios that were adapted from project A, which also show comments made by the consultant indicating usability problems.



Test Scenario		
Code	TC10	
Name	Perform a power reading using the camera and processing the image obtaining a success message.	
Summary	After accessing the reading types, the user may want to carry out a reading using the camera. At this moment, the application must turn on the camera. The AI service will process the images captured by the camera, detecting the meter and searching for reading numbers of power consumption. After identifying a valid reading, the system must lock the screen to the obtained image and register the numbers, confirming the reading with the user.	
Precondition	The user must be logged in the system and permission to access the camera must have been provided.	
Page for Starting the Interaction	Main menu.	
Steps	**User Actions**	**Expected Results**
1	Tap on the "Reading" option.	System will present the menu for types of reading.
2	Select "Camera" option.	System will turn on the camera and present the image being registered. The AI service will process the images.
3	Focus camera on meter numbers.	System will recognize a set of numbers, lock the image corresponding to the identified numbers and present the numbers on the screen. A confirmation button will be activated to save the reading.
4	Select "Confirm" option.	System will send image to server and success message will be presented. Message: The reading has been stored.
Results	Passed	
Doubts and Suggestions	During confirmation, how can the user cancel the action or restart a reading? Further details should be provided in the success message. For example: What was the stored number? In which account? In which Date? Perhaps the success message should be: A reading number has been successfully stored. Contract Number: XXXXX Reading Date: XX/XXXX Last Reading: XXXXX Updated Reading: XXXXX Obs. See corrections based on this test case in Figure 1 (Navigation and Information Redesign)	

Figure 2: Test scenario for a functionality from the mobile application and reporting of usability problems.

3.3 Project C - Automated Customer Support of Clients

Context: The goal of this project was to develop an information system to support the decision making process of prioritizing the contact of unsatisfied clients. In this sense, this application would provide relevant information on the dissatisfaction degree of a client based on information obtained from the data base in project A, and indicate which clients require immediate assistance.

Project Details: This project was carried out over a period of 18 months, starting in 2020 until 2022. In all, the development team consisted of 18 software engineers. The development team applied the same development process as in projects A and B. To define which deliveries would be produced and evaluated, goals were defined for each month regarding the development of each of the systems' modules implementing the solutions requested by the power company. The following activities were defined for the project: (a) Definition of scope and architecture (Months 1-3); (b) Database Development (Months 4-10); (c) Development of Report Generation and Decision Making Modules (Months 4-14); (d) Development of User Interfaces that would integrate data, make contact with users and produce reports (Months 8-18); and (e) Test, Integration and Release (Months 6-18).

Adaptations for Quality Assurance: In this project, the main issue was to track the different changes of all the modules that were being implemented. Also, there was a high degree of turnover in software developers, which caused trouble when maintaining current and previous software. As a result, the project manager decided to implement a branching model for Git, which is the version control system in use by our organization.

To decide which branching model would be employed, the management team, together with a subset of the more experienced developers, held several meetings with brainstorming sessions. In these meetings, we defined what risks/difficulties were frequent in projects A and B that could be avoided using a branching model to track changes. Among the main problems, the team highlighted: (a) difficulty in finding the correct files in the team directories; (b) difficulty in identifying who was responsible for modifying the artifacts; (c) difficulty in keeping information about results obtained in sprints in terms of developed features; and (d) difficulty in maintaining version control and quality assurance in terms of test execution.

After analyzing different branching models and considering the problems listed above, the development team decided to employ Gitflow, which is a heavily branch-based, but project focused deliverable model that defines the roles of each branch and how they should interact (Driessen, 2010). Among the branches used within Gitflow, the following are suggested (Olausson and Ehn, 2015): feature branches, release branches and maintenance branches. Gitflow has been adopted in several development projects with success but few practical information on how it was applied is provided (Bretal Ageitos, 2020). The team proposed a process in which the managers, developers and test team would have designated activities related to version control of the systems under development, following Gitflow suggestions adapted to the needs and reality of the team. To keep track of the various branches of the system under development, accounts were created in Git so that each team member was assigned the responsibilities of feeding the repository. In addition, codes and structures were designed to keep track of the actions of team members. For example, when creating a new branch of type *feature*, a developer would have to use the following standard: ”Create Branch - Feature - Feature Name”. Similar instructions and structures were adopted throughout the newly defined process.

4 RESULTS

In this section, we present how the processes applied in Section 3 yielded results for each project, and what lessons each team learned with these experiences.

4.1 Project A

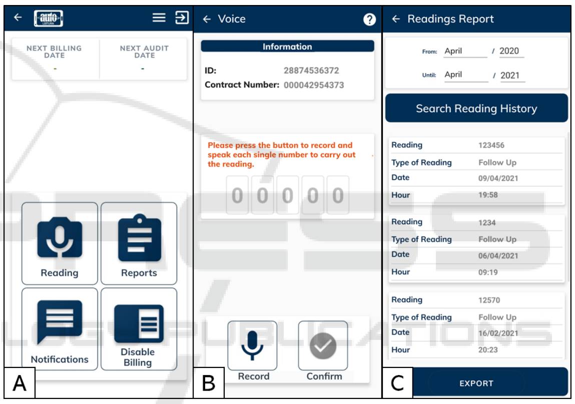
After identifying a problem, the development team adopted a report approach using a project management system[1] and a set of spreadsheets. In the management system, we indicated information such as: functionality id, tester name, problem description, team member that was responsible for the correction, degree of severity, date to correct the problem and an image of the problem. To decide which problems would be corrected first, a scale for the degree of severity was applied: (a) cosmetic: there is no need for immediate correction of the problem; (b) minor: it is a problem with low correction priority (could be corrected); (c) major: it is a problem with high correction priority (must be corrected); and (d) catastrophic: it is a problem of very high priority (if it is not corrected immediately, the system will stop working). This scale was useful for defining which problems were to be corrected and when. For instance, layout problems were considered cosmetic or minor if they not affected the usability of the system or were almost unnoticeable. On the other hand, miscalculations were treated as major or catastrophic problems if they impacted in the reliability of the software and if they where problems in functions with high degree of use from the point of view of users.

At the end of the project, 254 functional test scenarios were identified using exploratory testing. Through these tests, we verified the functionalities in terms of proper response of the system, adequate data presentation, system messages, input error handling and navigation. By the end of the project, all test scenarios passed and all minor, major and catastrophic problems were corrected, guaranteeing the proper functioning of the information system.

[1]http://trello.com

4.2 Project B

As in Project A, the functional testing through exploration allowed correcting issues with regards to wrong outputs of the system and missing information according to the requirements. Furthermore, we identified around 30 usability problems through the ad-hoc inspection. Figure 3 shows some of the screens in which usability problems have been identified. For instance, in part A (Home Screen) there is no information on the clients account and when (s)he needs to measure the power consumption. With regards to Figure 3 part B, messages were provided in orange, making the user think that an error had happened when, in fact, it was a message with instructions on how to use the system. Finally, in Figure 3 part C, there is a lack of information on loading time. If a report has a lot of registers, the interface will only load the report, which will freeze the screen.



ID: 28874536372
Contract Number: 000042954373

Please press the button to record and speak each single number to carry out the reading.

0 0 0 0 0

From: April / 2020
Until: April / 2021

Reading	123456
Type of Reading	Follow Up
Date	09/04/2021
Hour	19:58
Reading	1234
Type of Reading	Follow Up
Date	06/04/2021
Hour	09:19
Reading	12570
Type of Reading	Follow Up
Date	16/02/2021
Hour	20:23

Figure 3: Screens of the mobile application that contain usability problems.

The complete evaluation of the mobile application allowed us to identify improvement opportunities from both functional and usability perspectives. Figure 4 shows an example of the new version of the mobile application, after correcting the identified usability problems. In these screens, a new circle appears to indicate that the record has started so that users are aware of state changes within the application (see part A). Additionally, message colors were changed and in-between processing messages were added (see Part B). Finally, we standardized the reading screens and improved the layouts, so that users do not have difficulty when understanding different screens during readings (see Part C). These changes were made by considering the feedback of the inspector and also the feedback of the managers from the power company who received the evaluation report and prototypes of the corrected issues.

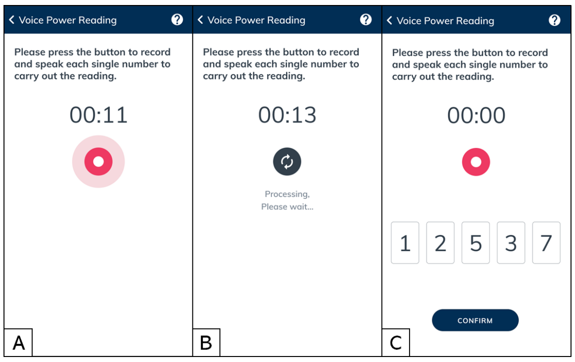

Voice Power Reading
Please press the button to record and speak each single number to carry out the reading.
00:11
A
Voice Power Reading
Please press the button to record and speak each single number to carry out the reading.
00:13
Processing, Please wait...
B
Voice Power Reading
Please press the button to record and speak each single number to carry out the reading.
00:00
1 2 5 3 7
CONFIRM
C

Figure 4: Redesigned version of the application - interaction of the meter reading through voice.

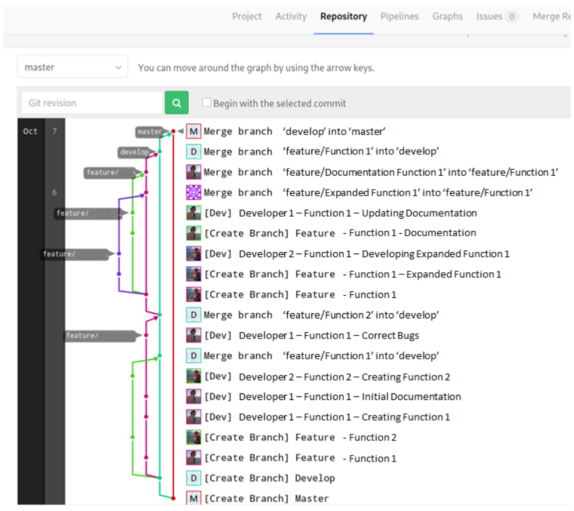
4.3 Project C

The proposed data registration templates allowed us to register the developed features within the given time stamp, the current version of the system, registration of the necessary configurations to use a module, instructions for Building and Deployment, as well as links with access to important external information.

With regards to the use of Gitflow, Figure 5 shows an example of how the team registered team members' activities in the Git version control tool. In this example, two developers are registering their activities when working alone in a branch or collaboratively. Initially, *master* and *develop* create their corresponding branches, initializing the project. From this point on, developers can work on separate feature branches to be added to the *develop* branch. When a merge with the *develop* branch occurs, developers talk to each other to resolve conflicts. In addition, when two developers work together on a branch of a specific functionality, a specific nomenclature was adopted indicating who is working on which part of the functionality, allowing a better control of the activities of the team members. At the end of the process, merges occur in the branches *feature*, *develop* and *master*, respectively.

5 LESSONS LEARNED

We were able to obtain lessons learned regarding the application of functional and usability evaluation activities as well as the version control activities. These lessons are mainly based on focus group meetings and discussions held with the development team of each project at the end of the reaching of a project milestone. We highlight that the discussion was held by team members who participated in all projects, while also considering the opinion of software engineers



You can move around the graph by using the arrow keys.

Begin with the selected commit

```
M Merge branch 'develop' into 'master'
D Merge branch 'feature/Function 1' into 'develop'
  Merge branch 'feature/Documentation Function 1' into 'feature/Function 1'
  Merge branch 'feature/Expanded Function 1' into 'feature/Function 1'
  [Dev] Developer 1 – Function 1 – Updating Documentation
  [Create Branch] Feature - Function 1 - Documentation
  [Dev] Developer 2 – Function 1 – Developing Expanded Function 1
  [Create Branch] Feature - Function 1 – Expanded Function 1
  [Create Branch] Feature - Function 1
D Merge branch 'feature/Function 2' into 'develop'
  [Dev] Developer 1 – Function 1 – Correct Bugs
D Merge branch 'feature/Function 1' into 'develop'
  [Dev] Developer 2 – Function 2 – Creating Function 2
  [Dev] Developer 1 – Function 1 – Initial Documentation
  [Dev] Developer 1 – Function 1 – Creating Function 1
  [Create Branch] Feature - Function 2
  [Create Branch] Feature - Function 1
D [Create Branch] Develop
M [Create Branch] Master
```

Figure 5: History of implemented collaboration using Gitflow within Project C.

who provided feedback during their participation in any of the projects.

The tests were affected by the *"communication within the project environment"*. A lot of information regarding business rules and function specifications was tacit (e.g. not documented, not thoroughly described or not stored in the same place), which made it difficult to identify test scenarios after the modeling of the system was completed. Exploratory testing allowed us to both understand the business rules and revisit decisions that had been taken.

The view from the product owners also affected the way in which the data would be presented. Several changes were made to the design of the application, including: new steps for performing tasks, new data analysis and filtering, changes in report presentation, layout and interaction. The Variability of the Functional Requirements affected the initially defined test scenarios. Thus, the changes must be monitored in order to guarantee that the correct expected result is being provided by the system. Each decision was reported to both the development team and the testing team during daily meetings. Also, the changes and their impact were discussed with the clients so that changes in the final product would reflect the power supply company's expectations.

Exploratory testing can benefit from *considering different types of test approaches*. To select alternative scenarios for designing test scenarios in the exploratory testing, the development team applied the use of functional testing strategies such as (Jovanović, 2006): equivalence partitioning, boundary value analysis and cause-effect analysis. Furthermore, although we initially tried to automate the test scenarios to facilitate their replication in later sprints of the project, since the discussions with the clients included dramatic changes in the interface, the development team had to change the testing strategy, using a manual exploratory testing. This made the functional testing process take longer than expected, but saved time with: (a) rework in terms of programming new test scenarios for the interface, and (b) not spending time on automating test scenarios that would be removed.

We also gained insight on the use of usability ad-hoc inspection for identifying problems and improvement opportunities. When combining exploratory testing and ad-hoc usability inspection, an experienced inspector can reduce the time necessary to check for different types of defects. Also, by simulating different usage scenarios, the test team can verify to what extent these scenarios meet usability principals, mainly when making mistakes during the interaction, or changing between tasks and navigating through the application.

When analyzing the *disadvantages in the application of the methods*, we were only able to carry out the evaluation due to the experience of the usability consultant. Not all development teams may have such experienced team members. However, by investing some time and applying an appropriate checklist with usability principals, an inexperienced inspector may be able to identify similar problems. Furthermore, the team had to deal with the delay of the identification and correction of the problems, as we carried them out after the exploratory testing. Although the identification of usability problems and improvement suggestions were welcome, these changes implicated in further development time for correction, and another round of testing and validation. In future projects, the execution of evaluation approaches for different software quality attributes should be executed together if possible, to avoid such degree of rework. Another alternative, given the appropriate human resources, would be to carry out the usability evaluation during the development of low fidelity or functional prototypes. This, however, was not our case, as the validation with the client was performed later in the project due to the clients' schedule, which also delayed the execution of the usability evaluation.

Finally, although Gitflow is still under testing in the development project, there was positive feedback. Regarding the *perceived usefulness* of the approach, there were comments related to the implemented organization, while obtaining the necessary information for project management. Furthermore, there were criticisms about learning the necessary steps to apply Gitflow. The project manager indicated that this difficulty can be negative in the short term, but that, as part of the organizational culture, it can bring benefits in the long term for the maintenance of projects. His main concern was regarding the required training and time spent for specific developers to play the roles of master and develop, as these branches have high credentials and errors could be dangerous to the consistency of the project's data and code. Furthermore, regarding possible difficulties and suggestions for improving the adopted approach, the team members initially indicated the adaptation to the process, due to the work required to learn the technologies and rules, as its use may be difficult or discourage developers with little experience. Additionally, the automation of some activities of the process was suggested (e.g. automated creation of tags to report changes) to improve productivity.

6 CONCLUSIONS AND FUTURE WORK

The applied software engineering approaches focused on two quality aspects within the software organization: product quality and project management. With regards to the testing approaches focusing on product quality, we: (a) adopted management tools and artifacts to report the identified defects; (b) used collaboration tools that allowed us to set tasks, organize the different states of a bug (e.g. pass/failed, reported, corrected, validated), made notes and use reminders; and (c) developed specific spreadsheets and documents as a way to keep track of the different decisions made, and which functional test scenarios were removed, updated or created. As a result, exploratory testing allowed us to create updated documentation that could be used for further stages in the development project, such as creating user manuals and project reports. Additionally, the ad-hoc inspection process allowed us to identify further problems from the point of view of users, which were not the focus of the previous performed tests.

With regards to Gitflow adoption for project management, we noticed the acceptance of the proposed process for quality assurance due to the positive feedback of the team members and that they are currently using it in the project to better control its versions. However, there was a unanimous concern in the short term regarding the ease of use and learning of the various rules necessary for adopting the approach. Considering these issues, the organization is studying new alternatives to automate or at least facilitate the use of standard messages and steps.

One of the main limitations from our experience reports is regarding the observations presented within this paper. We gathered information from the records of our daily meetings, discussions held during development, developed artifacts (documentation and reports) and the obtained results measuring the success of activities (e.g. number of identified problems). However, there is still room for improvement, applying questionnaires and interviews with the team members and carrying out focus groups with the development teams of the three projects aiming at gathering specific data on the applicability of the quality assurance activities and their acceptance in the future. We intend to carry out further investigations so our lessons learned can be useful for novice software development teams willing to adapt their development processes to achieve better results in terms of process and product quality.

As new projects are developed within our software organization, we expect that further software engineering approaches and artifacts are incorporated and that the lessons learned from such experiences are reported considering both quantitative and qualitative analysis methods. Through the lessons learned within this paper and by providing details for its replication, we intend to encourage software companies to adapt software engineering approaches, cost-effectively improving the quality of information systems.

ACKNOWLEDGMENTS

This work was supported by SijurI project funded by Equatorial Energy under the Brazilian Electricity Regulatory Agency (ANEEL) P&D Program Grant number APLPED00044_PROJETOPED_0036_S01. Additionally, this work was supported by the Foundation for the Support of Research and Scientific Development of Maranhão (FAPEMA), the Coordination for the Improvement of Higher Education Personnel (CAPES) and the National Council for Scientific and Technological Development (CNPq).

REFERENCES

Bretal Ageitos, F. (2020). Project management platform for the identification of phytoplankton samples.

Collins, E. F. and de Lucena, V. F. (2012). Software test automation practices in agile development environment: An industry experience report. In *2012 7th International Workshop on Automation of Software Test (AST)*, pages 57–63. IEEE.

Damian, A. L., Marques, A. B., Silva, W., Barbosa, S. D. J., and Conte, T. (2020). Checklist-based techniques with gamification and traditional approaches for inspection of interaction models. volume 14, pages 358–368. IET.

de Carvalho, J. M. I., da Silva, T. S., and Silveira, M. S. (2016). Agile and ucd integration based on pre-development usability evaluations: An experience report. In *International Conference on Human-Computer Interaction*, pages 586–597. Springer.

Driessen, V. (2010). A successful git branching model.

Hassani-Alaoui, S., Cameron, A.-F., and Giannelia, T. (2020). "we use scrum, but...": Agile modifications and project success. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*.

Hron, M. and Obwegeser, N. (2018). Scrum in practice: an overview of scrum adaptations. In *Proceedings of the 51st Hawaii International Conference on System Sciences*.

Jovanović, I. (2006). Software testing methods and techniques. volume 30.

Kamei, F., Pinto, G., Cartaxo, B., and Vasconcelos, A. (2017). On the benefits/limitations of agile software development: an interview study with brazilian companies. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 154–159.

Olausson, M. and Ehn, J. (2015). Source control management. In *Continuous Delivery with Visual Studio ALM 2015*, pages 65–85. Springer.

Oprins, R. J., Frijns, H. A., and Stettina, C. J. (2019). Evolution of scrum transcending business domains and the future of agile project management. In *International Conference on Agile Software Development*, pages 244–259. Springer.

Plechawska-Wójcik, M., Mora, S., and Wójcik, Ł. (2013). Assessment of user experience with responsive web applications using expert method and cognitive walkthrough. ICEIS.

Quesada-López, C., Hernandez-Agüero, E., and Jenkins, M. (2019). Characterization of software testing practices: A replicated survey in costa rica. volume 7, pages 6–1.

Rothenberger, M. A., Kao, Y.-C., and Van Wassenhove, L. N. (2010). Total quality in software development: An empirical study of quality drivers and benefits in indian software projects. volume 47, pages 372–379. Elsevier.

Ruane, E., Smith, R., Bean, D., Tjalve, M., and Ventresque, A. (2020). Developing a conversational agent with a globally distributed team: an experience report. In *Proceedings of the 15th International Conference on Global Software Engineering*, pages 122–126.

Theocharis, G., Kuhrmann, M., Münch, J., and Diebold, P. (2015). Is water-scrum-fall reality? on the use of agile and traditional development practices. In *International Conference on Product-Focused Software Process Improvement*, pages 149–166. Springer.

Wonohardjo, E. P., Sunaryo, R. F., and Sudiyono, Y. (2019). A systematic review of scrum in software development. volume 3, pages 108–112.